OCCASION

This publication has been made available to the public on the occasion of the 50th anniversary of the United Nations Industrial Development Organisation.

DISCLAIMER

This document has been produced without formal United Nations editing. The designations employed and the presentation of the material in this document do not imply the expression of any opinion whatsoever on the part of the Secretariat of the United Nations Industrial Development Organization (UNIDO) concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries, or its economic system or degree of development. Designations such as "developed", "industrialized" and "developing" are intended for statistical convenience and do not necessarily express a judgment about the stage reached by a particular country or area in the development process. Mention of firm names or commercial products does not constitute an endorsement by UNIDO.

FAIR USE POLICY

Any part of this publication may be quoted and referenced for educational and research purposes without additional permission from UNIDO. However, those who make use of quoting and referencing this publication are requested to follow the Fair Use Policy of giving due credit to UNIDO.

CONTACT

Please contact publications@unido.org for further information concerning UNIDO publications.

For more information about UNIDO, please visit us at www.unido.org

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
Vienna International Centre, P.O. Box 300, 1400 Vienna, Austria
Tel: (+43-1) 26026-0 · www.unido.org · unido@unido.org

19840

Advances in Materials Technology: MONITOR

Issue 27/28 JULY 1992

INDUSTRIAL APPLICATIONS OF MODELLING AND SIMULATION

PREVIOUS ISSUES:

Issue No. 1	Steel
Issue No. 2	New Ceramics
Issue No. 3	Fibre Optics
Issue No. 4	Powder Metallurgy
Issue No. 5	Composites
Issue No. 6	Plastics
Issue No. 7	Aluminium Alloys
Issue No. 8	Materials Testing and Quality Control
Issue No. 9	Solar Cells Materials
Issue No. 10	Space-related Materials
Issue No. 11	High-Temperature Superconductive Materials
Issue No. 12	Materials for Cutting Tools
Issue No. 13	Materials for Packaging, Storage and Transportation
Issue No. 14	Industrial Sensors
Issue No. 15	Non-destructive Testing
Issue No. 16	Materials Developments in Selected Countries
Issue No. 17	Metal-matrix Composites
Issue No. 18	Plastics Recycling
Issue No. 19/20	Advanced Materials Technology: CAD/CAM Application
Issue No. 21	New Materials Technology and CIM
Issue No. 22	Powder Metallurgy
Issue No. 23	High-Temperature Ceramics
Issue No. 24/25	Surface Engineering
Issue No. 26	Reinforced Plastics

Dear Reader,

This is number 27/28 of UNIDO's state-of-the-art series in the field of materials entitled **ADVANCES IN MATERIALS TECHNOLOGY: MONITOR.** This issue is devoted to the subject of **INDUSTRIAL APPLICATIONS OF MODELLING AND SIMULATION.**

The main article for this Monitor was prepared by Mr. Janusz Niwinski and is followed by articles which appeared in the proceedings which were published after the "1991 Winter Simulation Conference" in December 1991 in Phoenix, Arizona, USA.

We invite our readers to share with us their experience related to any aspect of production and utilization of materials. Due to paucity of space and other reasons, we reserve the right to abridge the presentations or not publish them at all. We also would be happy to publish your forthcoming meetings, which have to reach us at least six months prior to the meeting.

Due to financial constraints we cannot accept new subscribers for the time being, and also ask for your understanding as we cannot send all the requested copies of back issues as many of the past Monitors are out of print. **We kindly ask our readers to inform us whether you are still interested in receiving the Monitor and whether your address is still correct.**

For the interest of those readers who may not know, UNIDO has inaugurated the **MARINE INDUSTRIAL TECHNOLOGY MONITOR.** The Microelectronics Monitor and the Genetic Engineering and Biotechnology Monitor are also being published. For more information please write to the Editor of the respective Monitor.

Technology Development and Promotion Division

CONTENTS

1. MODELLING AND SIMULATION FOR ADVANCED MATERIALS TECHNOLOGIES

by

Janusz Niwinski

1. INTRODUCTION

This study is an attempt to present actual shape of simulation, the advantages and disadvantages of using it and the trend in present and future developments done in this area. Our goal is to give readers better understanding of simulation, to show that it is not only a computer game or "Star Wars" animation, but that simulation is a serious, useful and very powerful tool too, which can and has to be used in the planning and optimizing of manufacturing systems and processes.

The introduction of electronic computers had the same impact on industrial relations as had the introduction of steam power at the turn of the 18th century and electric power and combustion engine at the turn of the 19th century. It has led to the new industrial revolution. Electronic computer is a tool, which equally creates and supports new manufacturing conditions, causing very serious changes in manufacturing and organization techniques.

Many companies are faced with necessity of changing their research, manufacturing, and marketing strategies today. The following developments that take place in today's market create severe problems for the traditional, on Fordismus and Taylorimus-based, manufacturing:

1. Demand for individual products

The growing competition and the oversupply of different wares lead to a certain sales resistance. There are so many products the customer can choose from that he grows fastidious. He has money, he can afford more and he wants to pick out something special. He looks for particular items and is no longer interested in uniform goods, which he has already seen at his neighbours. He wants to get special, individual things, designed according to his unique wishes and requirements. Such an item should be an exceptional one, with high quality but not much more expensive than the standard one. Such wishes, better - such requests, must lead to substantial changes not only in manufacturing techniques but also in manufacturing philosophy and strategy. The goods must be cheap, so they have to be produced in long series at the automated manufacturing lines. This results in uniformity that is not asked for. The solution is to be found in the flexible manufacturing of short series of apparently different products which differ in form, function, fitting, and outfit, but can be produced in almost the same manufacturing facilities as the uniform goods.

2. Demand for short delivery times

Since the market and, as a result, the manufacturing is consumer driven, the manufacturers have to take into account not only what clients want to have, but also when they want to have it. The delivery times and as a consequence the manufacturing lead times are extremely short. For example, a large German shoe manufacturer produces his wares with lead time of two weeks and minimal batch size of 12 pairs. This length of lead times is not determined by the technological conditions. It results from a given organization of the manufacturing. At the time, he has to reduce it, because his competitors offer any 2 pairs of shoes with one week delivery time. It is a standard example, showing the trend present in many factories. The manufacturers have to produce in Just-In-Time not only to reduce amount of Work-In-Process or storage place and costs but also to stay competitive in terms of batch size and delivery times.

3. Demand for new materials

There are three main factors which have the biggest impact on growing demand for new materials:

Miniaturization

With the slogan "small is beautiful" and progress in development done in the field of electronics grows the number of items which are many times smaller than their predecessors. But small is not only beautiful, it should be robust and reliable as well. This leads to a growing demand for new materials and technology for integrated circuit chips manufacturing, new glues for assembling technology, new plastics and fibres for different parts where metal cannot or should

not be used, new and better special-purpose alloys, etc. etc.

Fight for new customers make the look of the product more important than ever. Clients should be attracted through products which are made not only smaller but also more fashionable in form and colour. This means demand for new material that can be shaped in any form the designer wants without losing any of the mechanical properties and robustness, material which can be dyed to any colour which is preferred by the customers. Changes in product form have also a considerable impact on the parts which should fit in this casing. They have to be made of materials which allow many changes of the form and size without losing its robustness and functionality.

Environment movements

Growing environment consciousness of society lead to the search for new materials which could be manufactured with less or no harm to the environment on the one hand, and on the other could be recycled without creating wastes causing water, soil and air pollution. This trend has an impact on almost all manufacturing branches causing long and expensive development processes and the subsequently reshaping of manufacturing. That leads to search and use of new material, on the one hand, in final products, on the other, in manufacturing processes; which will cause the necessity of employing new manufacturing methods and the building of new, or the reconstructing of existing manufacturing facilities.

Demands for individual designed products together with shortening of manufacturing lots' sizes and use of new material and technology has a weighty impact on the technological and organization form of the manufacturing and its environment.

Highly specialized products and use of new highly sophisticated materials cause growth of complexity of manufacturing processes and the same of manufacturing systems and its components. Additionally to this development there is a necessity of using flexible manufacturing components; which are induced by the short batch size and broad variety of product types and variants. Complex machines are expensive, it means they have to be used extensively to earn the money invested. To produce in smooth, flow manner, the manufacturing facilities have to be properly planned. The planning process should take into account not only all the elements of the present manufacturing environment (products, demand, batch size, delivery terms) but also the possible changes in it and the trends in market development.

These factors have a large impact on the shape of planned system elements like:

- Facility layout;
- Layout of the transport system;
- Storage facilities;
- Control and maintenance systems;
- External supply, transportation and communication systems;
- Expansion possibilities;
- Possibilities of changing the product type;
- Technological processes and requirements of present and planned products;
- Batch size ...

Planning of the facility, with taking into account these and many other factory specific conditions, is a very complex process and takes a lot of time. Unfortunately, the time is often not available. Product life-cycle is very short, and the facility planning and erecting have to be as short as possible. Since traditional methods of facility planning cannot fulfil these requirements, there is a need to employ new techniques. A rapid development done in field of computer hardware and software caused real revolution in manufacturing, not only in facility controlling and monitoring but also in facility planning.

Sophisticated simulation systems support developing and testing of the most complex manufacturing system. The results of the possible decisions can be known long before the decisions are made, machines bought and facility built. It allows checking in a short time various facility variants, with production of different products in different batch sizes (down to batch size = 1), and shows the consequences of diverse disturbance of manufacturing processes (machine or transport system break, shortage in supplies, changing order priorities, etc., etc.) on the work of the system.

A proper utilization of complex facility, producing short batches (often in Just-In-Time) of highly sophisticated products, use of new and expensive materials and technologies is a very complicated task. Growing number of orders, products with short batch size (down to 1!), and short delivery times make planning and monitoring of manufacturing with only use of traditional MRP/MRP II methodology an almost impossible task. Any minor disturbance in manufacturing could lead to disaster. The complexity of the system and the speed of changes are too big to be overseen with use of methods developed for the other manufacturing conditions (large batches, long delivery times and small amount of different product types being in the system at the same time).

Simulation based scheduling can be an ideal controlling and monitoring tool for supervising such a flexible manufacturing system (but also the traditional ones). Simulation will help the production controller to react properly to dynamic changes in manufacturing environment and to solve many of the scheduling issues. In case of change in manufacturing environment, a number of various problem solutions can be checked and the best one employed. Using computer model of manufacturing system, a couple of alternatives of various manufacturing processes will be simulated and after analysis of the results the best one chosen for execution. Simulation supports observation of manufacturing processes in time scale, showing possible consequences of single decisions and at the same time avoiding the employing of wrong ones.

Again, the goal of this study is to present actual shape of simulation, the advantages and disadvantages of using it and the trend in present and future developments done in this area. Our goal is to give readers better understanding of

simulation, to show that it is not only a computer game or "Star Wars" animation, but that simulation is a serious, useful and very powerful tool too, which can and has to be used in the planning and optimizing of manufacturing systems and processes.

2. STRUCTURE OF THE PAPER

This paper intends to give an introduction to simulation techniques especially in connection with industrial application with pointing out the areas where the manufacturing and employing of new materials take place. It should exhibit the real complexity of simulation and take away the false image of these methods which connected simulation mainly with computer games and animation from science-fiction movies.

It is arranged in eight chapters with numbers from 3 to 10 and Appendices:

CHAPTER 3. WHAT IS SIMULATION?

Chapter 3 describes the character of simulation. On the basis of dictionary definition and commonly known examples from ordinary life, the features and principles of the term "simulation" are described. The difference between simulation using scaled model of a real-world system and computer-based simulation, using appropriate examples is shown.

CHAPTER 4. APPLICATION SCOPE OF SIMULATION

This chapter shows the application scope of simulation. It is difficult to name all the fields and application cases where simulation can be used. In this chapter, "a feeling" where it can be used has been attempted. On the basis of shown application multitude, definitions of simulation application fields (movement - logic simulation, discrete - continuous simulation) are given.

CHAPTER 5. APPLICATION EXAMPLES

In this chapter, a couple of applications where simulation has been used are presented. In these real examples, not only the problems and the worked-out solutions are shown but also the savings of costs and time are listed.

CHAPTER 6. ADVANTAGES AND DISADVANTAGES OF SIMULATION

The first part of this chapter shows problems associated with experiments conducted using real-world systems. As a solution the employment of simulation methods is proposed. In the following parts of this chapter, a detailed survey of advantages and disadvantages connected with use of simulation is presented.

CHAPTER 7. CHOOSING SIMULATION SOFTWARE

Assuming that the previous chapters have sold the reader the idea of using simulation for solving given problems, this chapter describes the criteria which should be checked during searching for an appropriate simulation system from the existing vast number of packages. Additionally, the types of available simulation tools are described and some suggestions and hints which could be useful during searching processes are given.

CHAPTER 8. CURRENT TRENDS AND FUTURE PERSPECTIVES OF SIMULATION

In this chapter, res arch and development trends in the simulation area are described. They cover a wide domain, e.g., improvement of speed, user interfaces and modelling methodology, dynamic, interactive animation and analysis, on-line and hardware-in-loop simulation, 3-D animation and many other research directions.

CHAPTER 9. IMPLICATIONS FOR DEVELOPED AND DEVELOPING COUNTRIES

This chapter describes the chances of using simulation and the impact it could have on development in three roups of countries - developed, developing and post-communist States. The application fields covered in this chapter are: facility planning and manufacturing optimizing.

CHAPTER 10. CLOSING REMARKS

This chapter contains personal remarks and ideas about chances and perspectives of simulation usage.

Acknowledgements

This contains thanks to all the persons and companies which supported the author in preparing and writing this report.

References

Lists the materials that have been used for preparing single chapters of this study.

Appendices

Appendices - the closing section of this study - contained in Part A, Directory of Simulation Software, in the form published in 1991 by Society of Computer Simulation (USA). This directory follows readings recommended for obtaining a broader-based survey of sophisticated simulation systems.

3. WHAT IS SIMULATION?

Simulation is an expression which is known, and used by many people, at least in a general way. There are many definitions of simulation in different books and dictionaries.

In Webster's Unabridged Dictionary, for example, there will be found the following definition of this term:

"Simulation" - from Latin "simulatio", a feigning.

1. The act of feigning; pretence;

2. False resemblance, as though imitation.

The Random House College Dictionary describes simulation as:

1. The act or process of pretending, feigning;

2. The assumption or imitation of a particular appearance or form.

Dictionary definitions suggest that simulation involves imitation or mimicking, giving the appearance or effect of ... or taking on the characteristics of reality, and that simulation may involve a sham object, or a counterfeit, or a replica, or a model. Simulation brings to mind the idea of pretending, or feigning. These definitions are perhaps too general, too universal for our "technical" purposes, but they support the initial, basic understanding of simulation's meaning.

As technical term, we define simulation as the imitation of the operation of the real-world process or system over time. By a real-world system we mean some part of the real world, which is of interest. The system may be natural or artificial, in existence presently or planned for the future.

For imitating - simulating - the behaviour of previous defined real-world system, its model has to be developed. The behaviour of this model as it evolves over time allows the analysis of modelled system.

Simulation is often misunderstood as visualization. Visualization is mainly a pure displaying of data representing actual status of the monitored system. It can be only employed in connection with an existing and working system. Simulation is a depiction, feigning of the system behaviour. It imitates the work of the system using its model, while visualization demonstrates the status or behaviour of an existing real-world system.

The basic elements of simulation enterprise together with their relations are described in Figure 1.

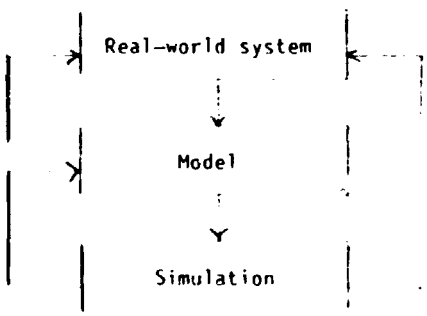


Figure 1

When an architect builds a three-dimensional model showing the proposed design of a building, the model is an imitation of the building. Easier, faster and much less expensive to build than the building itself, and much less costly to modify for the purpose of correcting mistakes or experimenting with possible changes in the design of the building, the model provides the architect with insights in terms of layout, feasibility, workability, and appearance. These same insights might not be available in two-dimensional drawings of the design (which are also simulations in their own right) or in attempts to visualize the design mentally. The model aids the architect himself or herself, provides the architect's client with an easily understood version of the design proposed for the building, and serves as a basis for communication (discussion) between the architect and client, making it more likely that each understand the other's intentions and that the two are in agreement on what the final outcome is to be.

The architect's model is an example of a physical, three-dimensional, and static simulation. The model is static (rather than dynamic) in the sense that the people do not move through the building or use its facilities, elevators in the building do not go up and down, the heating and air-conditioning systems do not work, and so on.

Examples of physical, three-dimensional, dynamic simulation models also come to mind not only by observing the current development in different areas, but also can be easily found in events and processes conducted in the past.

Henry Ford, determined to sell his Model T to average Americans, especially to millions of farmers, needed a plant which could produce his cars more effectively than the Highland Park facility. He believed that this problem could be solved only by employing a steady flow of materials and half-products through a manufacturing system.

In planning the great River Rouge plant, which displaced Highland Park in the 1920s as the heart of the Ford system, Ford insisted on having scale models of machine tools, conveyors, windows, pillars, and floor space, so that these could be moved around to test ideas about production. Between 1922 and 1926 he and his engineers designed and had constructed at the Rouge site a coke-oven plant, a foundry, a cement plant, an open-hearth steel plant, a motor-assembly building, and several other plants, which built one of the most important industrial complexes of its day.

The huge net of facilities at Highland Park and River Rouge with good organized, simulation-tested manufacturing allowed cost and time-effective mass-production of the Model T. In the five succeeding years of increasing production efficiency and savings, he cut the price of the basic car from $900 to $440, well below the price of the nearest comparable automobile. The average monthly number of unfilled orders grew to almost 60,000. In that time the Ford company had a 55 per cent share of the automobile market and production of the Model T climbed to 2 million cars and trucks a year.

Going back to the present time let us take, for instance, a major elevator manufacturer. He uses a physical model of a generalized elevator system in which elevators move up and down as time goes by. The user of this model decides how many floors and elevators to have in the system, sets the rates at which the people on various floors come to take an elevator, indicates the floors to which these people want to move, selects the rules for elevator movement (such as specifying that some elevators will only stop at floor 20 or higher), and then puts the model into operation, watching clusters of waiting people (represented by lights and counters) form on various floors, watching elevators pick up these people and deliver them to their destination, and obtaining statistics indicating the average waiting time, average time required for people to reach their

destination, and so on. By changing the arrival points and arrival rates of people and the number of elevators and the rules of people and number of elevators and the rules for their movement, the user can experiment informally with the model in "what if" fashion, searching for a design that balanced performance on the one hand with cost on the other.

The elevator model involves not just physical aspects, but also logical aspects. The experimenter provides information that the model uses, both logically and computationally, to imitate the elevation of an elevator system. Note that the model mimics not just the operation of the elevator system, but also reflects the passage of time. In the model, simulated time might go by more quickly than it would in a real elevator system. For example, it might only take 1 second for the simulated elevator system to make the moves that would require 5 seconds in a real elevator system. The effect of this time compression is to speed up the rate of experimentation, making operation of the model more time-efficient for the user.

Some dynamic simulations behave deterministically, whereas others contain probabilistic elements. In the model of an elevator system, for example, the time required for a moving elevator to travel between two consecutive floors might be some strictly determined value, such as 1 second. On the other hand, when an elevator stops at the lobby just before 8 a.m. on a workday, the number of people waiting for it might vary at random from simulated day to simulated day.

Simulation which is using only real, physical models of examined systems, as shown in the previous examples, has a limited scope of possible employment. It is difficult, or in many cases impossible, to use it for systems that cannot be described and built in the form of scaled models. An example for such a system can be weather model, the breaking out of a fire in a building, kinematics of parts during auto crashes, wider-seen space systems or power plant emergency cases.

The great change in spreading the scope of the simulation methods was caused by use of computer and computer-based simulation languages and tools. These computer-based simulations are not physical in the sense of a three-dimensional architectural model but can be physical in the sense that aspects of a situation are often represented pictorially on a two-dimensional screen. And, like the dynamic model of an elevator system, they can have logical and computational aspects, and inputs from the user. They often go further than this, providing challenging interplay with the user. For example, consider flight simulator, in which the user sits at a simulated control panel in the cockpit of a plane, with outside views shown through the windows of the cockpit. The user tries his or her skills at taking off, landing, climbing, diving, and banking and may crack up many times before developing the skills needed to control the plane successfully. After fundamental flying skills have been mastered, the user might then be ready to try dogfighting with enemy planes. This leads to a whole new set of learning experiments with a variety of dogfighting strategies to discover which ones seem to work and which ones do not.

4. THE APPLICATION SCOPE OF SIMULATION

It is very difficult to describe the full scope of simulation. Defining simulation as a feigning or imitation allows using its scope like a rubber band that can include almost all the areas of our activities. Reading a book we are simulating in our mind the actions and activities described in the text. Going to the movies or watching TV, we live lives of screen characters, feeling their loves, problems, stresses and thrills. Seeing the people leaving the movies, we notice women weeping after "Love Story", men walk like John Wayne after "Rio Bravo". We "simulate", we were the heroes of the movie. It can be said that we are living in a simulated world. But, let us leave the philosophical considerations.

In technique and science the scope of simulation use is very wide and steadily grows together with improvement of computer-based tools and methods. It's very difficult to show or list the full scope of simulation applications. Some of the situations that have been the subject of simulation-based investigations are listed in table 1.

Table 1

1. Aerospace - military - undersea

Space system reliability	Equipment replacement policies
War games/Strategies	Armed forces recruiting strategies
Search and rescue strategies	Equipment distribution
Space defence system	Satellite positioning
Combat vehicle training	Flight simulators
Radar and communication	Ballistics and missile systems
Navigation systems	Undersea cable development
Undersea vehicles	Target localization
Weather	

2. Health care

Health-care planning	Emergency room design
Organ transplantation strategies	Hospital staffing
Disease control strategies	Drug interaction control policies
Hospital admissions	Blood bank management
Diet management	Ambulance crew scheduling
Patient flow	Emergency situation planning

Table 1 (continued)

3. Urban-social

Emergency-response vehicle location
Traffic lights
Mass transportation systems
Air pollution control
Air traffic control
Urban development and dynamics
Garbage collection routings
Educational planning (schools buses)
Population planning
Weather
Airport and its environment design

4. Services and communication

Fast food facilities and nets
Local area networks
Supermarket planning
Telephone systems
Highway toll collection
Communication networks
Airport networks
Fleet scheduling
Harbour planning
Bank teller scheduling
Facility location and environment
Parcel service
Telephone switching
Freeway traffic

5. Politics and administration

Political redistricting
Economic condition
Portfolio management
Insurance and risk management
Labour planning
Political campaign strategies
Business games
Auditing strategies
Negotiation strategies

6. Industrial

Facility planning
Manufacturing optimizing
Repair and maintenance scheduling
Machine and tools monitoring
Quality control
Robots collision examination
Off-line programming
Control system design
Process scheduling
Inventory management
Distribution channels design
Process and product safety testing
Staff scheduling
Staff training
Power plant emergency strategies
Robot navigation

7. Product design

Engine design
Computer networks
Product testing
Computer and Integrated Circuit (IC) design
Nuclear and power plants
Product design
Design adaptation

8. Education and science

Heat and mass transfer
Hydraulics
Cardiovascular systems
Sonar systems
Soil and water systems
Chemical processes
Training
System dynamics
Cell growth simulation
Genetic
Isotope separation
Vibration
Propulsion systems
Educational simulators

The use of simulation is not restricted to one or several narrow classes of problems shown in table 1. This table shows just a small segment of possible applications. It should demonstrate the versatility of simulation and the possible range of applications and although suggestive, is by no means exhaustive. Just as a person can use simulation to experiment with flight simulator and try to get from Boston to Los Angeles: take off, fly the route, and land without a crash; to examine different routes, time of the day, weather conditions and flight strategies; so too can a person use simulation to examine the characteristics of proposed designs of a product or a system.

In the area of technical simulation applications, briefly described in table 1, there could be division in two sub-application fields:

1. Simulation of element movements in 3- or 2-dimensional space examines 2-D, 3-D relationships between system elements (robot simulation, Spacelab and Shuttle simulation, weather, flight simulator, etc.).

2. Simulation with primary goal on mathematical, logical or causal relationships between system elements (manufacturing, transport, urban systems, decision games, politics, etc., etc.).

In the following parts of this study, we will concentrate on the applications which belong to the second group of simulation employments; on the simulation systems and cases describing work of manufacturing processes and systems, their logic relations and behaviour in the time.

In literature we are often confronted with terms like continuous simulation and discrete simulation.

Continuous simulation describes systems using sets of mathematical equations. These may be algebraic or differential equations, usually with time as the independent variable. This kind of depiction can be employed for simple systems, which behaviour can be represented by such a mathematical function like fluid-flow, hydraulics problems, orbital calculations for a communication satellite or the representation of a blast furnace in a steel mill.

Discrete-event simulation describes a system in terms of logical relationships that cause changes of state at discrete points of time rather then continuously over time. Examples of such systems are manufacturing systems, queuing situations at gas stations, hospitals, etc. Discrete-event simulation assumes the lack of importance of things that might occur between modelling events. For example, examining manufacturing systems, we are interested in times where a given product arrives at the machine waiting queue and when it leaves it, not in what happens to this product between these points of time.

Today simulation systems, which we are interested in, are mainly discrete with the possibility to combine into discrete models with continuous methodology for solving of special class of problems and allowing fine-tuning of modelling of chosen system elements. In the following chapters we will understand simulation system as discrete-event system with possibility of continuous extension.

5. APPLICATION EXAMPLES

In this chapter, we want to show some applications where simulation has been successfully used. It is difficult to obtain the precise data about application and its results from companies which employ simulation. They consider this data as company classified information which could be used by competitors to harm the enterprise position. In description of the applications, names of the simulation packages used have been taken away. The purpose of the paper is not to make promotion for chosen simulation packages but in showing the fields and cases where these methods have been and/or are successfully used.

5.1 Total capacity management using simulation at Pratt & Whitney

In a project at Pratt & Whitney, a compressor blade manufacturing area which makes more than 50 parts was scheduled using simulation system. The area consists of a cropper, 6 extruder lines, 9 forge lines, 7 broachers and other stations that perform intermediate operations, such as machining, heat treat, and surface finishing. Material handling and storage is accomplished through integrated AS/RS and AGV systems. Altogether, the process includes 15 operations, with a manufacturing lead time of 8 to 12 weeks.

The broach is the bottleneck operation; its set-up takes from 1 day to 2 weeks and is a key consideration in production planning. Lots for the broach are typically sized at 10,000; whereas lots for the extrude and forge operations typically run around 2,000 to 2,500. The smaller lit sizes are designed to reduce inventory levels, while insuring the bottleneck operation has materials when needed.

The objectives of the scheduling project were:

1. Automation of routine scheduling decisions;
2. "What if" capabilities to evaluate scheduling decision alternatives;
3. Extend the scheduling horizon for manufacturing support organizations, tooling in particular;
4. Provide a capacity to evaluate reactions to unpianned events;
5. Provide a single coordinated schedule of all departments.

Tooling was one of the manufacturing area's largest problems. Even though large tool inventory was carried, tool-related production interrupts (wrong tools on hand) were experienced. Part of the long-range strategy is to provide sufficient forward visibility in the production schedule to support tool planning and scheduling. Purchased and fabricated tooling have lead times ranging from one week (expedited) to 6 months. In addition, forward visibility could benefit material purchasing since titanium stock lead time is about 16 weeks.

The blade area scheduling strategy used a combination of manual and computer-based steps. Each quarter, a schedule is manually developed for the broach, based on orders from the corporate MRP system. The plan horizon is 18 months, and accounts for part sequences and set-ups. From this, 18 month cropper-release schedule is manually prepared, using appropriate setback.

The simulation-based scheduling system is used to develop a 30-day schedule for the remaining operations. Scheduling for the two forge and two extrude operations required the consideration of a large number of capacity and operation constraints, and involved logic relations to intelligently sequence operation and plan changeovers.

The 30-day schedule is regenerated daily, using current status information from a CIM database. Production Control reviews the schedule to ensure tool availability and makes changes as appropriate. The schedule is then reviewed at the daily production meeting where further revisions may be made. Once accepted, the schedule is released to the production floor. Further adjustments are made manually. Simulation-based scheduling system incorporates these decisions in the next run. The 30-day schedule provides information to expedite needed tools within the 30-day window.

The next step is to expand the model-based 30-day schedule into the Disk Manufacturing and Experimental Blade area and work out a generator for 18 month broad schedule.

5.2 Car radiator core assembly line analysis

The manufacturer wanted to analyse the design of proposed aluminium radiator core assembly line to see if it would work as planned. They were concerned with understanding how the proposed system would behave with large variations in product mix and volume.

The simulation project had three main objectives:

1. Evaluate the proposed system design for flaws and inefficiencies;

2. Identify system bottlenecks that would prevent smooth operation;

3. Develop flow control strategies for system operation.

The project had been divided into two parts:

In part one, the analysis focused on refining the physical layout. Several improvements were recommended, including a change in the type of conveyor system used to feed and return parts. This change alone saved thousands of dollars, since the new conveyor was much less expensive.

The second part of the analysis focused on operational control. Flow control logic was developed to efficiently set up machines and release pallets to the assembly line for core production. This logic maximized production of the line while providing a smooth flow of finished cores to downstream operations. This logic from the model was eventually used to operate the completed system at two plant sites.

The analysis done in this simulation study both refined the physical layout and produced a sophisticated flow control logic system for this radiator core assembly line. Over $100,000 in conveyors alone was saved by implementing the altered system design suggested by the analysis.

5.3 Validation of assembly plant design

The automobile manufacturer was preparing to retool one of its assembly plants for a new type of car. Design engineers were preparing bid packages for equipment vendors and needed to validate the proposed assembly conveyor system.

The simulation study addressed three specific questions:

1. Can the proposed conveyor systems support the production objectives?

2. Could any excess conveyor capacity be eliminated?

3. What production rates were required for key assembly operations to meet production objectives?

The simulation model identified that, in order to meet the goal of 72 jobs per hour, part of the conveyor system would need to be increased. These increases were necessary to cover operations during breakdowns. The model also showed that other accumulation buffers were over-designed to cover operations breakdowns.

The simulation model showed that modifications to a proposed conveyor system for a redesigned body shop were required to meet the shop's target production goal, the analysis also identified an excess number of carriers in the system. By eliminating these carriers alone approximately $225,000 could be saved.

5.4 Analysis of semi-automatic assembly system

Ford needed to evaluate a machine vendor's proposed design to ensure that the recommended system could perform as stated. The vendor specified two different line speed possibilities, but could not identify an optimal setting. The system was a combination of automatic and manual assembly stations utilizing a palletized, non-synchronous belt for part transfer from station to station.

Six main objectives were defined for the simulation study:

1. Validate the accuracy of the machine vendor's proposal;

2. Determine the optimal number of pallets for the system;

3. Improve the method of pallet route scheduling;

4. Uncover any bottlenecks in the system;

5. Specify the belt line speed required to achieve optimal throughput;

6. Justify the project to division management.

A preliminary model was developed by Ford engineers that utilized a complicated automatic station with approximately 85 per cent reliability. Based on the model, they concluded that the system could not reach the desired throughput because of the limitation imposed by the inefficient automatic station. The automatic unit was replaced by a manual station, and the simulation was run again. The new results were extremely close to the desired outcome. Further simulations were run with differing combinations of belt speed and number of pallets to determine the maximum possible throughput of the system.

Ford Electronics and Refrigeration saved over $77,000 in facility building costs by using simulation to study a proposed semi-automatic assembly system. The analysis also allowed engineers to achieve the highest possible throughput by determining the proper line speed of the system.

5.5 Analysis of semiconductor wafer fabrication facility

Simulation has been used to solve problems in a large semiconductor wafer fabrication facility. Due to the high level of complexity in the manufacturing process, an accurate method did not exist to identify capacity constraints in advance of their occurrence. Also, when bottlenecks

occurred there was no method of accurately analysing the problem and identifying the most cost-effective solution. Many times, very expensive equipment (>$1,000,000) would be purchased to solve a capacity problem. Because the proper tools were not available to properly analyse the cause of the problem, decisions were made on very limited information. Many times, the equipment that was purchased would not solve the problem.

To solve these problems a simulation tool was implemented. It provides a productive way to analyse the various areas within the wafer fabrication facility. With the use of it, production personnel can now:

1. Predict production performance relative to expected demands;

2. Test the results of decisions (i.e. equipment purchase) before committing capital and resources.

With the help of simulation, highly accurate, complex production flow analysis is completed in a matter of minutes, bottlenecks are identified and "what if" analysis is performed to identify the most cost-effective solution.

The simulation tool implementation has provided the manufacturer with the visibility and accuracy to make better decisions involving capacity planning and short-interval scheduling and sequencing. Large cost savings have been realized due to the cost avoidance of several very expensive pieces of equipment. Several bottlenecks were minimized due to the ability to:

1. Identify potential problems in advance; and

2. Test the result of a proposed solution before committing capital and resources.

The implementation of simulation tool has resulted in:

1. Increased production throughput;

2. Decreased capital expenditures; and

3. Improved resource utilization.

Simulation is being used to help a large semiconductor manufacturer reduce costs by providing the ability to identify capacity constraints and test proposed solution before committing capital and resources. Simulation model has been integrated to Consilium's COMETS to provide on-demand analysis. The model is currently being extended to provide daily production schedules.

5.6 Final assembly scheduling for automotive engine cooling units

A division of a major automobile producer uses simulation tool to bridge the gap between existing planning system that provides assembly requirements in weekly buckets, and the operational necessity of meeting daily customer shipping requirements. The division manufactures about 75 different engine cooling units, that are assembled by a crew of 18 to 24 assemblers performing serial operations. There are typically 100 to 150 different orders in a system for a given week. Missed shipment can result in assembly plant downtime, incurring costs of about $100,000 per hour. Minimal amount of finished goods inventory can be held, so a missed ship date will require the use of premium shipping modes, costing the division as much as $100,000 per month.

The simulation schedules are driven by the daily-bucketed customer shipping requirements. The task of scheduling the final assembly area is complicated by the transition of the facility from a traditional assembly line layout to a more flexible cellular assembly configuration. The original three assembly lines are being replaced with 18 assembly cells, leading to a level of complexity that requires the computer-aided decision support provided by simulation.

The primary objective is to provide scheduling decision support that allows the scheduler to produce achievable, capacity constrained schedules that:

1. Meet customer requirements on ship dates;

2. Minimize final assembly inventory levels;

3. Maximize final assembly throughput.

A simulation-based scheduling system was developed to provide a sequence for each final assembly resource (cell or assembly line). The sequencing logic of the system evaluates the trade-offs between due date performance and changeover costs (as measured by the time required to make changeovers). The system captures all key constraints on assembly activities to ensure that feasibility schedules are produced. Final assembly schedules are produced with a one-day (four shift) horizon and a full-week horizon for distribution to the shop floor. Reports of component part requirements for departments that supply final assembly are generated by the system as well. Anticipated benefits from implementing a simulation-based planning system include:

1. Improved performance to customer ship date;

2. Reduction in finished goods inventories;

3. Improved communication between Manufacturing and Production Control;

4. Added visibili via simulation's "what if" capability, of the impact of alternative scheduling decisions.

5.7 Caterpillar Incorporated

Simulation system was used to describe the transportation of parts by vans and flatbeds from a central warehouse to the user buildings at Caterpillar's Aurora, IL assembly facility. Manpower and traffic pattern changes were analysed by using simulation tools to see how the transportation system will decrease and eventually disappear as they go to a Just-In-Time (JIT) environment.

5.8 Davidson Instrument Panel Division

A simulation model consisted of slush, mould, coating, and foaming facilities. The model enabled better optimization facility and manpower

requirements. The interaction resulting from the simulation project directed company capital expenditures and planned labour. By studying the dynamic interaction of the working model factory staff were able to plan the production output before any actual products were manufactured.

5.9 Johnsons Controls Incorporated

Simulation tools were used to model a Just-In-Time delivery system of automotive seats to an assembly plant. The simulation model helped plant location (distance), number of pallets, number of trailers, number of drivers and tractors, and requirements needed for finished seat set storage.

5.10 Battelle Memorial Laboratories

The objective was to investigate operational performance of an existing product line and evaluate design improvements. Specifically, the throughput of a 35-operation product was estimated and opportunities for potential reduction in manufacturing lead times and Work-In-Process inventories.

5.11 Touche Ross and Associates

This project consisted of simulation model to analyse various alternatives for a forging facility utilizing simulation experiments to consolidate and reconfigure multiple plant operations. Effective use of simulation models was the cornerstone of the manufacturing restructuring project.

5.12 Hewlett-Packard

Simulation was used to improve manufacturing efficiency of a production line with 16 work stations performing assembly operations. Through the use of simulation models, processes were streamlined to minimize Work-In-Process, manpower requirements were determined, and system flexibility and material handling system size were increased.

5.13 Rockwell International

Simulation was used as a capacity engineering tool for modelling of both the processes portion and electronics assembly portion of various production programmes. The objective of these simulations was to develop a strategy for factory automation, which would increase production output and reduce throughput time by implementing the most cost-effective automation solution.

6. ADVANTAGES AND DISADVANTAGES OF SIMULATION

Having shown some examples of the use of simulation, we comment in this section on the advantages and disadvantages of two approaches. Firstly, let us briefly consider experiments that can be performed using real-world systems; and then the pros and cons concerning the conducting of experiments with models of the real-world systems - employing simulation.

6.1 Disadvantages of experimenting with real-world systems

Having defined simulation as feigning or mimicking the behaviour of a real-world system, we must also say that even the best feigning stays only feigning and the best results in analysis of a given system can be reached by experimenting with the real system, which is the major advantage of this approach. However, it also has numerous disadvantages. Among these are the following:

1. The real system must exist before experiments can be performed on it, whereas the objective might be to design a system that does not yet exist.

2. If the system does exist and is in use, then for economic and/or political reasons it might not be feasible to interrupt its ongoing use for the purpose of experimentation. For example, if a manufacturing system is being used to build products, then it might be cost prohibitive to interrupt the production process to investigate the effects of making one or more changes in the system purposely.

3. Even if the real system can be used for experimentation, large amounts of time are usually needed to carry out the experimentation. If an existing system must operate for days or weeks while it is being observed experimentally after a change has been applied to it, then perhaps only one or two alternatives can be investigated at most, and even then the results of the experimentation might not be available in timely fashion. The same problems occur when the activities in an examined system are too fast, or too complex. It is almost impossible to properly monitor and analyse processes in integrated circuits. Millions of operations are taking place there every second, which allows only the observation and analysis of trends and not the sequence of single events.

4. Additionally, many systems are so complex, that the finding of the right data to be monitored, and where the monitoring could take place is a problem of its own, and not seldom bigger than the questions connected with the task to solve.

Simulation as a competing method has, of course, its own advantages and disadvantages. Here are some of them. Let us start with advantages.

6.2 Advantages of simulation

6.2.1 Realism of experiments

Simulation models can be realistic, not only in the sense of capturing the actual characteristics of the system being modelled, but also showing in an almost realistic form the work and reactions of the modelled system. The rapid development of computers, especially in computing speed and graphic capabilities, allows movie-like animation of the examined system.

6.2.2 Examining non-existent systems

Since simulation uses in its experiments not the real system, but its, usually computer based, depiction - a model, the system whose behaviour

has to be analysed can be also an imaginary one. It only needs to exist in the mind of the designer. A computer model of a planned system can be investigated exactly in the same way as the depiction of an existing one. This feature is very useful for the relatively inexpensive design of, and experimentation with products, facilities, or the whole systems.

6.2.3 Conducting experiments in time-scale

Many experiments which are possible to carry out using real systems or processes are very often too fast or too slow to make a proper observation, data collection or analysis. A complex process in a microprocessor with millions of operations and interactions of different elements takes only milliseconds. To monitor a real manufacturing system and collect an appropriate amount of data an analyser needs weeks or months.

Time can be compressed or stretched in simulation models. The equivalent of milliseconds and seconds in real system can be simulated in minutes or hours on a computer. On the other hand the equivalent of days, weeks, and months of real-system operation often can be simulated in only seconds, minutes or hours on a computer. This means that, relative to real-system experimentation, a large number of simulated alternatives of a slow system can be investigated, and results can be made available soon enough, or a detailed monitoring of fast processes is possible to influence its design or analysis.

6.2.4 Experimental control

In simulation experiments, every variable can be held constant except the ones whose influence is being studied. It allows reduction of "noise data" which do not have an impact on the experiment results, but in a real system have to be monitored. As a result, the possible effect of uncontrolled variables on system behaviour need not be taken into account, as must often be done when experiments are performed on a real system. Additionally every variable, also an artificial one which does not occur in a real system but which has an impact on understanding and performance of simulation study, can be set up and be monitored and collected during experiments thus improving results of the analysis.

6.2.5 Reproducibility of experiment conditions

In a real or imaginary system there are two kinds of events that can occur. They are deterministic events which occur at previously defined or known points of time and stochastic events that occur randomly with a given or assumed distribution. Because of true randomness of events in a real system, it is almost impossible to repeat a real-world experiment with the same characteristics and conditions. In an artificial, mathematical world, there are no true random values or systems. They are replaced, or substituted with pseudorandom numbers that are generated using a given mathematic formula. Which means that they are repeatable. Simulation experiments of such real-world systems are composed of elements that exhibit pseudorandom behaviour, and allow reproducing of real-world random events which, although perfectly predictable because of the pseudorandomness, can otherwise exhibit the characteristics of truly random numbers. This feature, together with the previously described variance reduction techniques and time compression give a possibility of repeating exactly the same experiment while focusing on different parts of the system. It improves the precision with which the characteristics of an examined system can be estimated and allow complete analysis of very complex systems or processes.

6.2.6 Training

Since the simulation is based on feigning, mimicking of depicted systems, it can communicate with the user using language of simulated system and does not require great level of scientific sophistication to be properly employed. This makes it easy to train simulation practitioners, and use simulation as a training tool. Simulation, especially with use of sophisticated computer medium, does not require from the user other knowledge than he or she has using a real system. This way, after a short introduction time it is possible to replace costly training done with real world system or processes through simulation-based training courses.

6.2.7 Winning over the client

Simulation is an ideal marketing tool. Because its concept, based on depiction of simulated systems is fairly easily and almost intuitively understood, clients are likely to be more receptive to the use of simulation than to descriptive depiction of the proposed system. As a result, clients are more likely to be inclined to implement simulation-based recommendations than those resulting from the use of models based on mathematics, functional diagrams, or 2-D drawings that they may not understand and whose results they may not trust.

6.2.8 Inexpensive insurance

It has been estimated that comprehensive simulation studies designed to estimate the characteristics of a proposed system can cost 2 per cent or less of the capital outlay involved in building the system. For example, it might cost $50,000 or less for simulation studies designed to evaluate a manufacturing system involving a capital outlay of $1,000,000. In this sense, simulation provides inexpensive insurance against building systems that are underdesigned and so will not perform to specifications, or that are overdesigned and so expensively provide more capacity than needed.

6.3 "Disadvantages" of simulation

Along with its many advantages, simulation is subject to some disadvantages. Looking at the disadvantages that can be found in all the simulation literature a little bit more carefully and meticulously, one will notice that these disadvantages are not the disadvantages of simulation enterprise alone, they are disadvantages (and pitfalls) of using any system. This part of the chapter will be taken as a kind of polemics with "disadvantage of simulation", that can be found in the basic work on simulation "An Introduction to Simulation using GPSS/H" by T. J. Schriber, Professor and Chairman of Computer and Information Systems in the Graduate School of Business at the University of Michigan, published 1991 by John Wiley and Sons.

6.3.1 Failure to produce exact results

Suppose a system is composed of one or more elements that are subject to random behaviour. In a hospital system, for example, the time required by a doctor to examine a patient may vary at random. The various times required by a doctor to examine patients influence the waiting-time experiences of others waiting to be examined by the doctor. When a simulation is performed with a model of the system, the values of such variables as "the time a patient spends waiting to see the doctor" are recorded by the model, and the averages of these values are given in a post-simulation report. But the average in a sample of observed waiting times only provides an estimate of the expected (or long-run average) time that patients must spend waiting to see a doctor. In this sense, a simulation only provides estimates, not exact results.

Simulation as depiction of a given real-world system is only a feigning of behaviour of that system. This means, it can be in extreme cases a very reflection of the system, but it cannot be better than the system itself. If a real-world system consists of a set of random events, it cannot be predictable. For example, the measurements made in hospital in month A differ from the measurements in the month B and it is impossible to make an exact prediction for month C. The randomness only of one of the system parameters is the cause of the impossibility of making an unequivocal description of the system. Without the exact depiction of the real-world system, there cannot be an exact model of it, and simulation producing exact data describing its behaviour.

6.3.2 Lack of generality of results

Simulation results apply only to the situations that were simulated and do not lend themselves to generalization. For example, suppose as it assumed in a manufacturing system, that manufacturing resources include three machines of type A, five machines of type B, and two machines of type C. Suppose that a simulation study is performed on this basis, and the resulting manufacturing rate is estimated. What manufacturing rate will result if there is one less type B machine or if one of the type B machines is replaced with another type C machine? The results from the simulation study already performed cannot be used to answer these questions. Instead, the model had to be modified to correspond to these changed conditions, and then simulation studies must be performed with the modified models to estimate the resulting manufacturing rates.

By the way of contrast, suppose a mathematical model has been built to express in the form of an equation or equations the manufacturing rate as a function of the number of type A, B and C machines. This model could be evaluated quickly and easily for all combinations of type A, B and C machines that might be of interest.

Generalization in manufacturing is a very sensitive problem. Through the high complexity of manufacturing systems and manufacturing processes, a slight change in parameters can have a large impact on system performance. For example, reducing a number of machines type A building a bottleneck in a system can lead to manufacturing disaster, while changes in the over-dimensioned type B have a slight impact on system work. In this case the lack of generality can be seen as advantage of simulation not allowing drawing false conclusions.

Changing number and kind of resources (machines) in a previously described manufacturing system can lead to completely new manufacturing configuration and can be often complicated. Changing a model does not have to be so complex. Defining machines as "independent" modules, we can simply replace one through another.

Mathematical modelling can seldom describe the complexity of manufacturing systems with all the dependencies not only between the machines and other hardware systems components but also between these elements and products that have to be produced on modelled facility.

6.3.3 Failure to optimize

Simulation is used to answer the questions of the "what if" type, but not of the "what's the best" type. In this sense, simulation is not an optimization technique. Consider the type of manufacturing system question posed earlier: "What combination of product should we produce if the objective is to maximize profit subject to the following machine, manpower, and marketing constraints?" Simulation can be used to estimate profit that will result when a given combination of products is produced. That involves a "what if" situation. In other words: "What if we produce this combination of products? What profit will result?" But simulation cannot be used to indicate which combination of products among all feasible combinations results in the maximum profit. This would involve a "what's the best" situation. In other words, "What combination of products is best in the sense of maximizing profit?" In simulation, the only alternatives considered are those that are directly investigated. Simulation does not generate solution; it only evaluates those that have been proposed. If six alternative combinations of products are investigated, then that alternative among the six considered that maximizes profit can be identified, but it is quite possible that one or more of the alternatives not considered result in larger profits than the best of the six that are considered.

Simulation alone is not an optimizing tool but can be a part of an optimizing system. In our example, we are looking for optimal product combinations for a given manufacturing facility, a simulation experiment will be conducted using as input a sequence of products to be made. After the experiment, input sequence together with the output parameter like system utilization, lead times, manufacturing costs, etc., are analysed by optimizing tool. It results in generating a proposal on new product sequence for the next simulation experiment. The new experiment will be performed, and then the analysis. Such a loop consisting of simulator and analysing (optimizing) algorithms can lead to working out at least a very good solution for the stated problem. Finding the real optimum for a usually very complicated manufacturing system is in most cases a Sisyphus task.

6.3.4 Long lead times

A simulation study cannot be conducted over a weekend. Months of effort can be required to gather data; build, verify, and validate models; design experiments, and evaluate and interpret the results. A simulation effort should be started well before the results are needed. In practice, unfortunately, the results of simulation are usually needed "yesterday". A simulation study may not be authorized until the project involving the system to be simulated becomes of urgent priority, and then there may not be adequate time to complete the study before the results are needed.

A simulation is a long process. It uses data describing a system, builds a model of it, performs a couple of experiments and analyses the results. For building a model, simulation professionals need a data about a modelled system. In many cases, that data is not available. The companies are working, in many cases, based not on reliable data but very often on the experience and feeling of its staff. There is never time to do measurements, complete and marshal the data. The analysis of the system has to be done, logical connections and operation times have to be determined. All the data should be obtainable a long time before the simulation study begins. But usually this is not the case. Searching data becomes a part of simulation study and prolongs the lead time of the study. Another very important factor of a simulation study is the cooperation with the examined system operating staff. In many cases, supporting the simulation team is only a side job with corresponding results. It is very difficult to define a simulation goal, obtain data, and check the model with the real system if there is not appropriate collaboration with company staff. This same problem arises during analysis of simulation results. What is important for the company, which parameters do not match the reality, etc. etc.? It is difficult to answer these questions being an outsider. The bottom line is that the long lead time is not a peculiar brand of the simulation, it is more the result of falsely prepared and conducted study on the part of supervisors of an examined company or systems.

6.3.5 Costs for providing a simulation capability

Establishing and maintaining a simulation capability involves making a major and ongoing commitment with concomitant personnel, software, hardware, training, and other support costs. Many smaller organizations cannot afford to maintain a simulation capability. Some organizations may have one or two people who work on simulation projects from time to time. Such people may have to be supplemented by outside consultants on occasions when simulation projects are to be conducted. Other organizations may simply contract out their occasional simulation projects to consultants or firms specializing in simulation. They will then pay a premium to have a simulation study conducted and may wind up in a position of dependency and relative inflexibility if follow-on simulation efforts are required.

Every small or medium sized company lives in a net of dependencies. It is very rare when a firm is really independent. There is always a net of suppliers, dealers, contractors and sub-contractors. If a company wants to introduce a new product, it hires a consulting company to make a market research examining the chances the product has. Product design is often done by a specialized company which has experience in 3-D design (it is too expensive to maintain its own staff of designers with appropriate hard- and software). Marketing campaigns are almost always done by external marketing specialists. The taxes and payments are supervised by external experts and lawyers. All the external companies a firm is employing to carry out jobs create a dependency relationship, and employing a consulting company for performing some simulation studies will not endanger the client situation.

6.3.6 Misuse of simulation

There are many facets to a balanced, comprehensive simulation study. As a result, a person should be educated in a variety of areas (e.g. analysis of input, design of experiment, analysis of output) before becoming a simulation practitioner. This fact is sometimes ignored, however, resulting in situations in which people who only know how to build simulation models and make runs with them are cast in the role of simulation professionals, even though their education and training may not have prepared them adequately for this designation or for these responsibilities. Such people may not be in a position to conduct balanced and comprehensive simulation studies. As a result, such studies may be incorrectly performed, or may be incomplete, or may fall short in other ways, perhaps resulting in failure of the simulation effort.

There is a need for understanding that simulation is not a game, it is a serious tool that can have a huge impact on decisions made. People are used to employing machines, and nobody would be set up as an operator without appropriate knowledge and experience. Nobody will be hired as manager without references and researches concerning his career. Why should a simulation tool be employed by staff without corresponding knowledge? If simulation is used by wrong people there could be a disaster, but the same, or similar disaster could take place if complicated and expensive machinery is exploited by the wrong people, or when the company is directed by the wrong managers. It is not only in simulation that misuse is a disadvantage, it is a disadvantage in any system that could be used in right or wrong ways.

In conclusion, simulation is not a panacea. It offers powerful advantages, but these advantages can only be utilized through a proper employment of this tool. Simulation has its own disadvantages, as every system has. Fortunately, most of the disadvantages connected with system performance and usefulness are diminishing in importance with time, thanks to improved simulation tools, methodology, education and computer performance, and decreasing computing costs. On the other hand many of the disadvantages imputed to simulation alone, are simply shortcomings connected with false employment or the misuse of any system.

7. CHOOSING SIMULATION SOFTWARE

Let us say that based on information given in previous chapters, or other informations, and

after analysis of your company and its needs you have decided to use simulation as a tool for solving some of your manufacturing issues. The choosing of proper simulation software from the vast number of packages available is a difficult task. In this chapter we want to introduce some elements which should be considered searching for a proper choice.

In the Directory of Simulation Software (1991) published by the Society of Computer Simulation (Appendix A), one could find 37 simulation packages that name manufacturing as one of their application fields. A person new to simulation could spend months looking at vendor literature and demonstration diskettes, and examining evaluation copies of software.

7.1 Classes of simulation software

We can distinguish four classes of simulation tools:

1. Spreadsheets;
2. Rapid modelling tools;
3. Simulators;
4. Simulation languages.

7.1.1 Spreadsheets

Spreadsheets are not generally known for their simulation capabilities, but it is possible to perform simple simulation using some of the spreadsheet functions or macros. Simulating with use of spreadsheet software takes a minimal amount of time, mainly because the systems are small or the applications are limited. When using a spreadsheet, the output is defined by the user. The output is usually in graphic form, with some very attractive results. Such attractive outputs are made possible by business graphic offered by many spreadsheets' packages. In short, spreadsheet simulation is used, pretty often without even using "simulation" term, for small systems, where the main emphasis is laid on the business graphic data presentation.

7.1.2 Rapid modelling tools

Rapid modelling tools (like ManuPlan II from Network Dynamics, Inc. Massachusetts, USA) are used to gain an idea about such measures of performance as throughput and bottlenecks. The system is modelled in very general terms, omitting many of the details in order to get an idea about the performance measures. In many instances this level of output is sufficient as it answers questions that are being asked in a timely manner.

7.1.3 Simulators

Simulators are data-driven systems that require no programming for certain types of models. A simulator consists of a couple of predefined functions, implemented as "black boxes" not requiring any or very small knowledge about modelling and simulation theory. These blocks, functions, can be set together in a sequence describing operation of the simulated system. The model scope is limited to use only the predefined set of blocks, provided with the simulator, and it is very difficult to find a way to trick out the system and go beyond the scope. However, some simulators allow the programming either within the simulator, dropping into another language, or through adding compiled code written in one of the standard programming languages like FORTRAN, C, or Pascal. The time required for modelling using a simulator is moderate, regardless of what the vendors say. Real problems are almost always more complex than the made-up examples that are used for teaching purposes. Simulators usually offer good output analysis. Built-in business graphic capabilities with predefined functions like pie charts, bar charts, or histograms allow direct output of simulation results on a laser printer. The file generators permit communication and data transfer between simulator model and spreadsheets or graphic software. A two- or three-day training course is usually provided by the vendors, and it is generally important to receive this training rather than to rely only on the documentation.

7.1.4 Simulation languages

Simulation languages are the most sophisticated and flexible simulation tool. Virtually any realistic problem can be modelled using a simulation language, although it may require a large investment of time to ensure that all of the details have been captured. This feature can be also seen as an advantage of simulation languages. Detailed analysis of modelled system prohibits the false use of simulation, forcing the modeller to detailed analysis of a given system and on the other hand allows very detailed modelling. Simulation languages offer a very wide spectrum of output features beginning at standard business graphic up to the user-defined output forms or on line communication and data transfer with other analysis tools. Three- to five-day basic training courses are usually offered either by the vendors or by third parties. To obtain more expertise, additional one- or two-day courses can be taken, after having worked with the software long enough to attack some real problems.

The distinction between simulators and simulation languages is blurring. Some of the simulation languages have moved towards the simulators by offering predefined block functions supporting fast, very often graphic, model development. On the other hand, simulators provide more and more programming possibilities for extending modelling flexibility. This makes the choice of propriety tool more difficult.

7.2 Selection criteria

Looking for simulation tool, as set up by Professor Jerry Banks from Industrial and Systems Engineering at the Georgia Institute of Technology, there are some features we classified in this study in the following groups:

1. Input features;
2. Processing features;
3. Output features;
4. Support and environment;
5. Cost benchmarks.

Words of warning are in order before you use the criteria shown in this section. First, you

must know which criteria are appropriate. Hence, it is not necessary to ask for 3-dimensional animation unless you know how you will use it. It is not necessary to ask for ten random number streams or ten distributions, etc., unless you know their purpose and that you will need them. The main thing is that you must understand the criteria and what is important to your situation. Second, the response to some of the criteria should not be judged on a "yes" (present) or "no" (absent) basis, but on the ability of the language or simulator to perform a service needed. A good example is the first criterion on the list, interface to other software (FORTRAN, C, Pascal, etc.). A simulation language or simulator should be judged on its ability to avoid the need for FORTRAN, etc., not simply whether it can be interfaced with a programming language.

The features discussed in this section are meant for application to simulators and simulation languages. Spreadsheets and rapid modelling tools are not meant for detailed simulation analysis so they should not be judged by these features.

7.2.1 Input features

Interface to other software packages

This ability is very important if we want our model to communicate with other software packages. For example, if our model of manufacturing facility should be fed by data coming from MRP/MRP II system in Manufacturing Planning Department, it will be unreasonable to relinquish that communication feature.

Input flexibility

A simulation system should allow all the input forms which can be in use for a specified group of planned and possible applications. It should be receptive to inputs provided:

- Interactively;
- In batch mode;
- In form of interaction with other hardware and software systems like Data Basis or Job Shop Data Collection System.

Input data analysis capability

Data serving direct as an input for simulation experiments does not have to be in pure form. In many cases, there is a necessity of having an input, for example, in form of exponential distributed pseudorandom values. Input data analysis should be understood as a possibility of using different preprocessing functions, properly preparing demanded values of the parameters.

Portability

A programme can be written on one class of computer and run on another class of computer. This option is a very important one. Choosing a proper simulation package, you can let it grow together with your applications and area where the simulation is used. Usually, simulation will be employed in the first step for a relatively simple problem. After the users notice the advantages of it, the application field grows together with demands for faster and more powerful machines for more complex simulation models and output analysis. Having a simulation system that is available on different hardware platforms, there should not be any problems with extending the operation field.

Syntax

Modelling terminology should be easily understood by the user and should be consistent and unambiguous. It should allow the user to communicate with the model using language he understands, using terms common in the application area, not particular computer-specific names and expressions. Variables, indexes and functions should be referenced using names not numbers, outputs should be descriptive etc., etc.

Interactive debugger

This option is especially important at the development stage of the simulation model. It gives a modeller control over execution of every single simulation step and gives him an access to any data set used in the system. Especially useful for such debugging purposes can be animation option used in parameter watch, trace or step by step mode.

Development tools for animation

Animation, often seen as only a game, is a very important feature in contemporary simulation experiments. To be properly used, this option should come together with some characteristics like:

- Ease development of animation picture;
- High quality of picture;
- Number of colours, picture size, minimum and maximum object size;
- Smoothness of movement;
- Portability of remote moving;
- Interface to standard graphic and CAD programmes;
- Zooming of the animation pictures.

7.2.2 Processing features

Execution speed

Speed is a factor the beginners often do not appreciate enough. For simple simulations, it can be overseen. With time, problems become more complex and the number of simulation runs necessary to get the proper result increases, it is very frustrating (and expensive) to have people sitting and waiting until the computer is ready with its job. It is important in constructing models since numerous runs are made for validation purposes. It is very important in making production runs consisting of many scenarios, each of which is replicated numerous times.

Model size

For PCs running under plain DOS, this factor is important. For computers running under Extended DOS, OS/2, VMS, or UNIX this factor is much less important.

Material handling features

If the scope of application needs material handling elements like transporters, conveyors, robots, or cranes, be sure that the appropriate elements are included in the investigated simulation tool.

Random variable generators

This option is usually overseen in the searching phase, but is very important for future exact modelling of investigated systems. It should include the basic distributions as exponential, uniform, triangular, and normal, plus empirical distributions.

Reset and restart

This feature allows the discarding of observations, saves the recorded observations and status of the model and restarts the experiments in the future beginning with the saved status. This possibility supports partitioning of long experiments into short independent parts, and having a look into the data collected.

Independent replications

This option describes ability to perform a set of experiments automatically using different sets of input data with collecting outputs from every run.

Data structures

A simulation tool should give a possibility to create a set of global data, which can be reachable and changeable for every entity moving in the system. Every entity should have a user defined data set (large enough) describing its properties. This local data can be changed at every point of simulation experiment according to alterations in entity properties.

7.2.3 Output features

Standardized reports

Standardized reports include performance measures such as average number in queue, average time in queue, average utilization of resources, entity lead time and throughput. This option includes also standard business graphic output forms like bar charts, pie charts, histogrammes and other plots that can be directly sent to a laser printer.

Customized reports and file creation

It is very important to have the possibility of creating different user-defined files. Format of these files should make possible using them in various databases and graphic packages employed in the company. One of the file formats that have to be available as output option is ASCII Norm. The ASCII text files can be accepted by almost every software package and represent the most universal readable communication form of data.

Database maintenance

Simulation requires numerous replications of one or more scenarios. This leads to necessity of managing, saving and analysing of huge amounts of data in a standardized and organized form. The only possibility of administering these quantities of data is by employing various databases. A simulation system can have its own database structures which can manage the saved data and/or must supply appropriate on-line connection to typical database systems.

7.2.4 Support and environment

A number of self-explanatory characteristics belong to the environment features. These features should be checked while searching for a proper simulation system. Criteria used while review of simulation packages should be set in accordance with the knowledge and experience of the staff that should use them. Among the check measures are the following:

- Ease of use;
- Ease of learning;
- Quality of documentation;
- On-line help;
- On-line tutorial;
- Customer support:
  - Training,
  - Technical support,
  - Updates and enhancements.

7.2.5 Cost benchmarks

Price is a very difficult factor to put a value on. There are packages from $1,500 to $80,000, but the software price does not say anything about the usefulness of the package for a given application. Depending on employment, different tools with different prices can be used. Software price alone does not cover all the expenses connected with a simulation tool. You should consider other costs connected with using a chosen package. They include hardware requirements, software and hardware maintenance costs, update and on-line consulting fees. Only the addition of software price to the "additional" costs connected with using simulation tool can give a real estimate of the price of the package.

7.3 Some general hints

Having checked the software features we would like to give some general hints towards helping to make a proper choice of simulation tool. Looking for an appropriate simulation tool, you should have in mind that every vendor is trying to sell his product and is using all the allowed means to attract the customer and show that his product is the best one of the 37 you find in the directory.

Get the greatest power that you can afford

Once a package has been selected, buy the fastest version of that software. Having simulation analysts wait for output is usually more costly than paying for more number-crunching capability.

Beware of fancy advertisements and demonstrations

This caution is especially true of some simulators. A simulator can have wonderful interfacing capabilities and produce attractive

outputs, but may have a very weak and less than robust engine. This is not an indictment of all simulators.

Beware of checklists

Beware of checklists that list generic features of simulation software. The presence of some of these features is not nearly as important as the implementation and extent of capability offered by the software overall.

Obtain a trial copy of the software

Many vendors offer a student version, a limited edition, or a trial copy of the software. It is advisable to obtain a copy of the software in this form, as well as the software documentation, and use it for solving a small version of the problem. It is still difficult to know the capability of the full version of the software until a real problem is tackled.

Ask the vendor to solve a sample problem

The problem would probably be a reduced version of the real problem which you are trying to solve. This request may involve a consulting fee, but it can have great payoff. If you are willing to put some money on the line, the vendor will know that you are serious. Otherwise, you should not expect a vendor to assign an analyst to a potential non-revenue activity. If the vendor cannot solve the sample problem to your satisfaction, then you should look at other software possibilities. The payoff is in avoiding the purchase of software that will not solve the type of problem you encounter.

You are going to buy a decent simulation language

The most popular simulation languages are all decent tools. Their features and performances may be quite different, but their robustness is impressive. You will not get stuck with a lemon.

Decide if you need animation

Animation is a very powerful tool, both in the developing phase as a debugger of model behaviour and during a simulation experiment. If you can afford it, take it. Many simulation packages have animation as an inexpensive option. They are worth buying.

Try an independent consultant

If the choice of simulation software is too complicated and you do not have enough time, experience or capacity for making a proper selection, there is always the possibility of using an independent third party to help with the decision. You can find a consulting company experienced in the field of simulation which can help you find a proper simulation package for either your single application, or wide application area. Using a consulting company for software choice services can be prolonged for training and trouble-shooting activities at the intitial stage of using simulation.

8. CURRENT TRENDS AND FUTURE PERSPECTIVES OF SIMULATION

Simulation is a useful tool in analysis and design of complex manufacturing systems because it can give insight into manpower and equipment requirements, cycle time, inventory, and throughput for alternative designs before any physical changes are made in hardware or facilities for a new system. A steadily growing number of problems that could be solved using simulation methods and the increasing interest for these techniques in the industrial world cause growing attempts to improve existing and develop new ways of simulation usage. The steadily growing number of industrial applications results in changing the image of simulation from a university, somewhat theoretical, game to a very useful analysis tool, which can be used in a wide area of industrial implementation.

The application related developments done in the simulation field (let us leave research conducted in the field of simulation methodology and theoretical studies) can be divided into the following groups:

- Modelling and problem representation;
- Input features;
- Run time, performance features;
- Analysis features;
- Animation.

8.1 Modelling and problem representation

A large part of the work in a simulation study consists of writing, testing, and debugging the simulation programme. Even though simulations are written in languages that are designed for simulation programming, they are like programmes written in any language in that they are time consuming and labour intensive to write, test, test and debug. This limits the applicability of computer simulation as a modelling and analysis tool. Complex simulation models must be built by simulation experts that are expensive to hire and hard to find. Currently, it is not easy for the average plant engineer to build a simulation model of a complex system. To help the end user in his attempts to build simulation models, much research is done. The trends in improving modelling of manufacturing systems go in the following directions:

8.1.1 Graphic interactive modelling

Many performance modelling systems are, or are just ported to be, graphically based, enabling the user to build a model by directly drawing and manipulating a pictorial model diagram on the computer display. These modelling systems should generally provide a menu driven interface and an icon palette with which the user specifies the model by selecting and linking icons and then providing associated textual attribute information. The resulting combination of graphics and text forms the model specification and requires that certain rules be obeyed in terms of content, completeness, syntax and semantics. Some classes of user errors can be completely prevented with this type of interface; others need to be detected and identified to the modeller, ideally as the model is constructed. Within the visual modelling, the detection and display of errors is significantly more complex than in text based modelling languages. Additionally, graphic modelling could free the user from the abstract

commands of textual modelling systems communicating with him using his own language; language that is closely connected with his real-world system and which is his natural one. He should not have to understand (translate) terms like Delay, Seize, Free or entity attributes. He takes an icon describing machine and lets it perform a given operation on a defined part.

Through the proper use of graphic interactive modelling which supports modelling done by the plant engineers the modelling cost and times can be considerably reduced. Plant engineers who can conduct their own simulation studies are able to obtain reliable data inexpensively. Their day-to-day experience in their plant allows them to validate models as they build them. The study can incorporte input from people in the plant at all stages while the model is built, validated, and exercised.

8.1.2 Simulation program generators

In order to increase the payoff from an investment in simulation modelling, one would like to be able to reuse the model to solve more than one problem or design more than one system. There is much research done in this area. Simulation models should be built in a way that supports the use of the parts of one model as segments in another one. This "exchange" should be done not only through "cut and paste" methods, commonly used in today's modelling, but by the employing of special (or general) purpose simulation program generators. Such a generator should allow the user to reuse pieces of simulation code and data structures by automatically aggregating those pieces together into a model, according to the user's specification. The purpose of a generator is not only the assembling together of different modules to form a logical and causal structure of a new model but also the controlling data communication interfaces between single parts of the model and error handling features in the validating and experimenting phase.

8.1.3 Object Oriented Simulation (OOS)

There are some developers that declare that the new simulation software should be object oriented. Object-oriented programming offers some interesting features, that could support more natural understanding of modelling processes and models itself. It is caused by the fact that the real world is object-oriented and our thinking is object oriented too. Object Oriented Simulation (OOS) should offer significant potential over existing popular simulation languages in several respects.

Reusability is perhaps the most convenient feature of Object Oriented Simulation. Object defined within an object-oriented language are inherently extensible. You can craft new objects out of existing ones. For example, an AGV object may be crafted from a fork-truck object, since many of the properties of the two are similar.

Object Oriented Simulations are modular, with objects being the modules. With modularity, all the information known about the objects is held in one place; you do not need special procedures to find information. This encapsulation of information means that changing the meaning of an object or modifying its behaviour is easy to do, and changes of the object can be easily maintained.

In short, the new object-oriented simulation languages should offer features like:

- Clear process/event structure;
- Efficient clock mechanism;
- Combined continuous/discrete modelling and;
- Inheritance.

It also should have an environment which supports graphics, interactive simulation and auxiliary modules (program generators) for queuing and continuous models.

8.1.4 Artificial intelligence

The major impact the Artificial Intelligence research could have on simulation is encouraging the use of additional kinds of modelling based on inferencing, reasoning, search methods, and representations that have been developed in AI. This natural, though long overdue, extension of simulation should produce behavioural models that answer questions beyond "what if ...?". The result is sometimes referred to as "Knowledge Based Simulation". The AI methods should explain why a given sequence of events occurred and answer definitive questions as "Can this event ever happen" or goal-directed questions such as "Which events might lead to this event?" expanding the number of problems where simulation can be employed and completing the picture of the behaviour of models of the real-world system, the way we are used to see the original systems.

8.2 Input features

Complex simulation experiments are often conducted by outside simulation experts who may not be available to help the plant engineers after the simulation study has been completed. The full potential of existing simulation models may not be realized if the models cannot be used by the plant people for carrying out their own experiments. They cannot use the model because the input and analysis interfaces are written for the simulation experts, not for the real-world system user. Another major limitation of input interfaces for simulation is that users such as plant people, managers and decision makers need to know too much.

The last development in the interface area tries to connect an abstract world of numbers used in simulation environments with meanings these numbers have in a real world, making communication between model and outer world simpler and comfortable.

Graphical user interfaces (GUIs) are prepared to support particular kinds of users in solving specific types of problems. These users are the plant engineer, decision maker or a manager, who typically is not involved in the development of the simulation model, and hence would be unlikely to know or want to know details of modelling, simulation, programming or query languages, or database structures. The advanced programming of graphic interfaces should allow a very fast depiction of the model data and structures in a form which can be accessible and understandable for a single user. With mouse and menu, he could communicate with the model of the examined system, set parameters, supply attribute values for objects, and animate a simulation run.

8.3 Run-time and performance features

By employing faster scientific workstations and increasing the execution speed of the examined models, the number of special features which could be implemented and used during simulation experiments will also grow.

8.3.1 Restart simulation from any time point

Research done in this area includes graphic querying of the simulation state, being able to roll the simulation back to a previous state, change a parameter, and rerun the simulation, saving multiple simulation states for later analysis, and comparison. It results in the possibility of saving the status of simulation experiments at many points of time and allowing the performing of a set of simulation runs with different parameters' values starting not from the beginning, but using previously conducted experiments. This development will have a big impact on the lead times of the simulation experiments and the same on the reduction study costs.

8.3.2 Changing sensitivity during experiments

Sensitivity analysis is very important for indicating which parameter and which parameter values are the most important to a given output function. The straightforward approach to sensitivity analysis requires running a simulation many times perturbing individual parameters to see how the results differ. In old simulation systems this process was prohibitively expensive in most cases, and as a consequence is rarely done.

The last developments, employing Artificial Intelligence methodology, are searching for the sensitive parameters and value boundaries, automatically changing input parameters and generating subsequent simulation runs. It should very fast provide the name of the parameters which have the biggest impact on the given goal function and where lay the appropriate value boundaries.

8.3.3 Aggregation

Another major direction of current simulation development is searching for the methods for varying the level at which they are aggregated (also referred to as their "resolution" or precision with which the model describes the real-world system). It is generally necessary to choose a desired level of aggregation in advance and design a simulation around that level. Changing this level typically requires considerable reprogramming of the simulation; changing it under user control or dynamically is generally unthinkable. The fact that the level of aggregation of a model gets frozen early in its design is a major impediment to the reusability of simulation in general.

Last developments, which employ Artificial Intelligence methods at the modelling stage, should allow the user the varying of level of aggregation of a simulation during executing simulation experiments and to indicate which aspects of the model are of particular interest, running those aspects of simulation disaggregated while running peripheral aspects at a higher level of aggregation.

8.3.4 On-line simulation

One of the most important research areas is an attempt to use simulation as an on-line-tool. It should support the monitoring of an existing real-world system and employing simulation as a trouble-shooting tool. A simulation model should run in real time with the depicted system, comparing at given time points the status of a model with the status of a real system. In case of minor differences the model alters its status automatically. In case of considerable differences, like a machine break, changing manufacturing priority, lack of materials, a series of fast simulation experiments controlled by an expert system will be started with purpose to work out an acceptable solution for a given problem.

To be able to perform such purposes, new simulation systems have to be developed. These systems (languages) allow constant communication with the outer world. In these systems events in the outer world and events in a model have the same impact on a model behaviour and permit uninterrupted interaction between both worlds. The research done in this area will support very complicated scheduling processes and permit implementation of real Just-In-Time manufacturing.

8.3.5 Hardware-in-loop simulation

Since the real-world system becomes very complex and expensive, there should be the possibility of examining parts of the system. The studying of the system components should allow not only checking behaviour of the component as a closed entity but also testing the interaction of its elements with the system for which the component is planned to be employed. In this area, which is a kind of on-line simulation, much development is conducted. A large system will be depicted in the form of a simulation model. A small part of it, for example a machine, a transportation system, a control system for storage facility, exists in reality. The model runs parallel in real time with real component of a system supplying it with the signals which describe model (representation of the whole systems, excluding the real component) behaviour. Output signals from the real component are read by the model and used for succeeding simulation steps. This way, using appropriate simulation systems, it should be possible to examine the interaction of various system components without building in reality the whole system.

8.3.6 Using distributed, parallel computing

Sequential calculating is a traditional way of solving problems with the help of computers. All the traditional simulation systems work this way. The development of transputers opens new horizons for speeding the execution of simulation experiments. Using a transputer extension card with 8 transputers supported by their own operation memory chips should theoretically improve the performance speed up to 8 times. The developments done in the area of distributed simulation go in two directions:

Porting the traditional simulation languages in such a way that the same model with different sets of parameters could be executed on more transputers. This feature is especially

interesting for different optimizing algorithms, where a number of simulation runs have to be performed to find a maximum (minimum) of a given function.

The fields where such parallel systems can be used are for example:

- Scheduling of manufacturing processes;
- Search after the best system configuration;
- Trouble-shooting in on-line simulation;
- Hardware-in-loop application for fast systems;
- Checking sensitivity and aggregation of simulation models;
- Animation of complicated graphic systems, etc.

Parallelization of simulation models. This feature should allow the cutting of a model (automatically during execution time) into more parts, which can be executed parallel on different processors. The parallelization of simulation experiments, and use of inexpensive transputer cards allow the solving of many complicated and time-expensive problems on low-priced PCs or workstations instead of using sophisticated and expensive high-end computers.

8.4 Output analysis

It could be said that the simulation is as good and useful as the conclusion drawn after or during experiments. With proliferation of user friendly interfaces grows the number of developments concentrated on supporting analysis activities. The developments are mainly concentrated in the following fields:

8.4.1 Interactive graphics

There is a lot of work done at the time in the field of interactive graphics. Plots should respond to user input, using a command-style or mouse-drive interface. A good example is point identification, where the analyst uses the mouse cursor to select a point in the diagram, causing a label identifying the associated case. Another example of interactive graphics is zooming and scrolling of diagrams allowing monitoring of chosen scale areas independently from the actual time points. These options should make analysis of simulation experiments more organic and natural. Making analysis in the real world, there is usually a possibility of focusing on a chosen part of the data or observing and comparing data originating from different points of time.

8.4.2 Dynamic diagrams

Research in this field is closely coupled with the interactive graphic. Plots and graphics change over time, responding to changes in the examined system showing the actual status of chosen variables. Connection of interaction and dynamic of the output should make on-line analysis of simulation outputs simple and comfortable. Free scrolling through the time without losing data, which is not visible on the screen should speed up the analysis and make it more easy for use by staff without a sophisticated computer background.

8.4.3 Communication

Much work is done to accomplish integration of simulation outputs with various databases and software packages in one net. It opens simulation data for external analysis and presentation. On-line communication with external soft- or hardware systems allows efficient monitoring and controlling of simulation experiments. This way simulation data may be analysed by staff not prepared to use simulation, but used to put these sophisticated data analysis tools into operation. The profits of this configuration are eminent:

- Simulation lead times are shorter;
- Analysis of simulation output is done in the form used in the company;
- There is a growth of simulation acceptance (It delivers readable and usable data!).

Using on-line communication can lead to shorter decision times in case of on-line simulation; output data is instantly available at the place where the decision is made and in the form decision makers are used to.

8.5 Animation

With the appearance of faster scientific workstations, researchers may now obtain real-time animation of simulated systems with good graphics rendering for the objects involved. This represents a significant advance for scientists since they now can expect excellent visualization capabilities for physical systems under study. New and inexpensive hard- and software graphic capabilities are the cause that development in animation area has become one of the main research fields in the simulation area. These developments are concentrated mainly in the following directions:

8.5.1 Animation speed

Through the employment of special purpose graphic cards and corresponding development of simulation software, the speed and quality of the animation increases. The growth of graphic speed supports the use of animation in on-line real-time simulation. The improved performance of computer graphic cards allows the employment of a more realistic animation picture, use of more colours, more detailed depiction of real-system representations and smooth movement of animation elements. It also has an impact on use of various user friendly characteristics like scrolling, switching, zooming and recording animation pictures without impact on the simulation performance.

8.5.2 Communication

The work conducted in this area should allow recording simulation experiments for future use. It should be performed through connecting animation conducted on a computer monitor to a VCR or movie camera which could record the experiments. Simulation "movies" can be used later for external analysis of simulation results

(dynamic diagrams) and can also be used as a marketing tool showing different variants of proposed facility.

8.5.3 Interactive animation

Much of the work is done to allow improved user communication with the animated simulation. The user should be given the possibility of "talking" with animation using mouse and pull-down menus. He could change animation speed, stop simulation, change parameters' values and restart the simulation experiment, scroll animation picture in case it is bigger than computer screen, zoom a chosen part of animation picture, change between different pictures and layout parts of simulated systems.

8.5.4 Incorporating physical parameters

The objects are given mass and inertia, then forces and torques are applied to achieve a certain motion, which usually looks very realistic. While these systems certainly represent the future, they are still in their experimental stages, for it is often awkward for a human to specify the animation in terms of physical parameters.

8.5.5 3-dimensional animation

With growing computer performance and improvement in both graphic hardware and software systems there is a possibility of employing 3-dimensional animation. This time 3-D animations are used mainly for analysis of robot animation and various flight, cockpit or weather simulators. The 3-D simulation systems are expensive and so is the hardware used for this form of graphic representation. With the fall of computer prices and progress in software development, there is a strong trend toward employing 3-D animation for discrete-event simulation.

9. IMPLICATIONS FOR DEVELOPED AND DEVELOPING COUNTRIES

In this study we investigate the possibilities of using discrete (discrete/continuous) simulation in two main areas:

- Planning of manufacturing facilities; and
- Planning, monitoring and optimizing of manufacturing processes.

In this chapter we want to examine by implication what the use of simulation methods has and could have in the future in countries in different stages of development.

Through the changes which took place in the last years, we can distinguish three groups of states:

- Highly developed countries;
- Post-communistic countries;
- Developing countries.

Each group of these countries has its own specific attitude towards simulation methods, what has an impact on the scale of development, applications and overall understanding and acceptance.

9.1 Highly developed countries

This group consists of highly developed and industrialized countries like USA, Germany, Great Britain, Japan, or France. These countries are the main suppliers of industrial products which are consumed in the world. In these countries simulation is a fully recognized and utilized tool. The main development done in the simulation area is conducted there. There is a strong net of simulation societies supporting development and application of simulation tools. Most of the simulation employments come from these countries. Strong and expanding companies use simulation as a medium which helps them stay competent. The growing number of simulation employments leads to growing investment in simulation development, which again makes simulation systems faster, user friendlier and more widely usable.

In the group of developed countries the United States of America plays the leading part. In the Directory of Simulation Software published by the Society of Computer Simulation, there were 92 different simulation systems offered in the year 1990 and 116 in the year 1991. Summer Simulation Conference 1991 published proceedings with 1,186 pages of highly sophisticated papers presented in 16 different subject groups. Winter Simulation Conference 1991 comes with 1,235 pages of papers published and 11 groups. This shows that developing and applying simulation is a field with much activity. This shows also that simulation is a method that helps saving (and earning) money and time.

At this time simulation is mainly used in planning of manufacturing facilities. Almost all manufacturing and assembly systems by General Motors or Siemens are planned using simulation methods. Not only facility layout and utilization of the machines are examined there with the use of discrete/continuous simulation. Also the paths of AGVs and movable facility elements are checked (collision investigation and path optimization) using discrete/continuous/3-dimensional simulation.

The strong trend can be seen towards use of simulation methods for the scheduling of manufacturing processes for both traditional long batch, and flexible Just-In-Time manufacturing.

Simulation is for sure one of the factors that made the industry of these countries more flexible and competitive, and its application field grows steadily. There has been built a very effective circulation - industry uses simulation and believes in it - it brings money to universities and software developers active in simulation research - according to industry wishes, they develop new, more effective simulation systems - these systems are used in industry and make money - industry supports simulation research - and so on. It will be very hard for other countries to compete with such a system without using this same methodology.

9.2 Post-communistic countries

This group of countries emerges after severe economic crack-down and overthrowing of

communistic regimes in Central and East Europe. This group consists of States like Poland, Hungary or Czechoslovakia together with the new States, which were parts of the previous Soviet Union. At the time a very complicated process is taking place in most of these countries. They try to transform the State-owned centrally-managed industries to private free-market enterprises. The economic reforms connected with slashing subsidiaries for inefficient government-run factories, transportation, trade and other branches of the national economy should lead to improved effectiveness of work of these enterprises. The companies should be independent or privatized and work based not on central government-steered regulations but on free-market rules.

These reforms change not only the supervisor-company relations but also have a significant impact on manufacturing and organization techniques in the companies. It could be said that the overthrowing of the inefficient economic system left a large amount of different hardware (machines, transport systems, communication nets, etc., etc.) in an organizational vacuum. There is a large number of companies with highly qualified staff, in many cases very sophisticated equipment but without adequate products to be made, without sufficient organizational structure, which should include not only the managing of direct manufacturing but also take care of research and development, market studies, sales and promotion and all the other fields which are necessary to back up factory existence and expansion.

One of the examples of such enterprises is the whole military complex. The end of the "cold war" and the US-Soviet arms-control deal results in the necessity of converting over 2,500 plants in the former Soviet Union alone from military to civil manufacturing. The converting of the chosen 430 factories should cost from $30,000,000,000 to $40,000,000,000 and demands a very strong technical and organizational assistance.

In such a situation, the use of simulation based tools is not only a possibility but rather a necessity. The building up of new manufacturing structures together with the creating of new organizational forms on such a scale is not possible to manage and supervise with only the use of traditional methods. Additionally, time is a very important factor. Factories cannot stand still waiting while the solutions are being worked out. The economy cannot be stopped. There is a need of fast smooth transition from present chaos to well-organized structures of the manufacturing, transportation, communication, trade and administration.

In this group of countries, simulation tools are not well known and were very rarely used, mainly on a small scale at the universities. To support this technology which is necessary to make a fast transformation of the economy a number of promotion activities is required. Within the framework of such promotion and information studies, the following tasks should be performed.

- A broad introduction to simulation methodology;
- Presentation of successful employments of simulation tools;
- Presentation of simulation at the universities;
- Organizing simulation courses with use of various simulation systems;
- Independent supporting selecting simulation software;
- Giving assistance and supervising projects implemented with use of simulation;
- Supporting and establishing a net of companies active in the area of simulation development and consulting.

All these activities should lead to broad acceptance and implantation of simulation techniques necessary to reconstruct companies and to make them more competitive. But simulation is not only the tool for planning and organizing manufacturing facilities. The models used for planning, after slight modification, can also be used for the monitoring, controlling and optimizing of manufacturing processes.

As a bottom line, it must be said that simulation is the only possible solution for making the right about-turn from being very static, closed, State-owned and controlled to a free-market, dynamic and competitive economy. Without simulation it would be a very taut and long process with strikes and social turmoils.

On the other hand, simulation is the only tool which can support the complicated processes necessary to manufacture new sophisticated materials and products made by them; and no manufacturer who is not capable of using sophisticated simulation-based tools can be successful and competitive.

9.3 Developing countries

They are nations where the largest part of the world population lives. They have the richest raw material deposits but are mainly poorly industrialized. The incomes in these countries are usually low in every respect: total and per capita. Many of them sell raw material and agricultural products, importing industrial products. Since the prices for raw materials and agricultural products fall down they have also trade deficits. Probably, the way to solve the economical problems of these countries is through the industrialization of the national economy. Raw materials should be processed into the final products in the country, saving money, creating jobs and improving living conditions of the population and raising the technical, cultural and scientific level of the country.

It could be said that these countries are at the start position. The factories have to be built, economy, transportation, communication and trade organized, in reality, from scratch.

Of course, industrialization is a very complicated task which, in the case of developing countries, cannot be performed without the assistance of international organizations, or industrialized, developed countries. There is not only the necessity for new machines and technologies but also for new, contemporary organization techniques.

Simulation is one of the techniques which could be quickly implemented and used. Being an imitation of the real-world behaviour and communicating with the customer using language he is used to employ, simulation can be utilized very fast and in natural form by the user without sophisticated industrial knowledge and experience.

Surely, simulation has to be introduced and supported by organization patronizing industrialization of the country, but it is one of the conditions which will decide about success or fall of development programs. It is not enough to install machines; what is more important is the proper organization of the factory, the proper utilization of these machines and their proper maintenance and control and these goals can be reached using simulation base planning, controlling, monitoring and maintenance tools.

Since the industrial and scientific structure of the developing countries is very sparse, a program must be started which will introduce and support simulation techniques. It should support the following activities:

- A broad introduction to simulation methodology;
- Presentation of successful employment of simulation tools;
- Starting studies and research in the simulation area at the universities;
- Organizing simulation courses with use of various simulation systems;
- Independent supporting selecting simulation software;
- Giving assistance and supervising projects implemented with use of simulation;
- Supporting establishing of a net of companies active in the area of simulation development and consulting.

These activities should be more strongly supported and supervised as in the case of post-communistic countries. The reason for this is that many developed countries suffer from lack of sufficient academic and scientific structures and institutions and shortage of qualified research and development staff.

Fields where the developing countries could become leading manufacturers are:

- New material research and manufacturing;
- Production of goods based on domestic material resources;
- Processing of agricultural products;
- Small, manpower intensive manufacturing.

In these areas, resources which are available in these countries could be used without the necessity of importing large amounts of foreign materials. In such cases what they need is technology and the organization of manufacturing.

As a bottom line, it has to be said that simulation will have a growing importance in development and management of different fields of life in all of the countries. In some of them (developed countries) it is already an appreciated and widely used tool; in others (post-communistic and developing countries), it should be supported by some sponsors (UNIDO, universities, international industry, software companies, government, etc., etc.). But, there is no way into the future which can be used without employing these techniques.

10. CLOSING REMARKS

Simulation, and especially computer based simulation, is for sure one of the tools which will have a growing impact on the development of manufacturing techniques. The main advantage of simulation lay, from its definition, in its ability to experiment with a model of a studied system, instead of examining the real system itself. The growing costs of manufacturing systems components and the shortening of planning and developing times make experimenting with the real-systems very complicated, if not impossible.

Additionally, the ability to work with an **imaginary system** or processes, which exist just in the heads of developers makes defining of the future simulation area a very difficult task. In any case, the growing number of successful simulation employments led to propagation of simulation methodology and creates new customers. This proliferation of the simulation idea together with developments which try to make simulation more user friendly, faster, more effective and natural in its way of work will lead to new look of the modelling and simulation enterprise.

The growing importance of **new materials** which manufacturing and processing require employment of very complex manufacturing processes and highly sophisticated machines can be efficiently planned, monitored and optimized only with use of computer based simulation tools.

Additionally, the necessity of employing **Just-In-Time** manufacturing which is characterized by short batch, large variety of different types of products being manufactured at the same time in flexible manufacturing systems, and short delivery times lead to very complex manufacturing. Planning, monitoring and optimizing of such manufacturing is almost impossible using traditional tools developed for big batch manufacturing where the production can be planned years before and where the manufacturing facilities are built to make only one type of product.

Observing developments occurring in market and manufacturing situations, it can be said that there is no way into the future which could pass by the use of simulation techniques. It could be that the used ones will get another name, for example forecast systems, but the employed techniques stay the same.

For sure, simulation cannot and will not stay the exclusive domain of developed countries, which use simulation in any possible area. If the post-communistic and developed countries want to improve their manufacturing and overall organization of the national economy they have to learn, appreciate and employ the simulation techniques. These processes have to be supported

by sponsors like: international organizations, universities, international industry, software companies, government, etc., etc. Without the assistance and without use of sophisticated simulation tools, there is small chance for reaching the manufacturing and organizational standards which are common in developed countries.

At the end, it could be said that in the simulation field we are at the same point the computer development was in the late 1970s and early 1980s. That was the time of change from large mysterious batch-mode main frame computers like IBM 704 with their priests and air conditioned rooms; through Altair and Apple II to the omnipresent Personal Computers and Workstations with their user friendly Graphical User Interfaces.

In the next few years simulation will become omnipotent in every process and development saving time, money and resources. And not only by Giants like General Motors, Ford or Siemens but also in small companies as a planning, monitoring, controlling and optimizing tool.

11. ACKNOWLEDGEMENTS

It is a pleasure to thank the many individuals who deserve recognition for their help in the writing of this study. I would like to give special thanks to Tom J. Schriber, Professor and Chairman of Computer and Information Systems in the Graduate School of Business at the University of Michigan and Jerry Banks, Professor of Industrial and System Engineering at the Georgia Institute of Technology for their support and works used in preparing this study; Dave W. Kelton, Professor in the Department of Operations and Management Science of the Carlson School of Management at the University of Minnesota for his help in preparing the recommended readings; John Hammann President of System Modelling Corporation, Karilyn Griggs from CACI, Jenny Mishler from Pritsker Corporation and Ina Bonnen from AT&T ISTEL for supporting this study with necessary information materials and case studies. Finally I would like to thank Pablo Spinadel, international expert for industrial automation for his constructive suggestions during the writing of this study.

REFERENCES

3. What is simulation?

1. Schriber, Thomas J. 1991. An Introduction to Simulation Using GPPS/H. New York, NY., John Wiley & Sons.

4. Zeigler, Bernard P. 1985. Theory of Modelling and Simulation. Malabar, FL. Robert E. Krieger Publishing Company.

6. Hughes, Thomas P. 1989. American Genesis. New York, NY. Penguin Books.

4. Application scope of simulation

1. Scriber, Thomas J. 1991. An Introduction to Simulation Using GPPS/H. New York, NY. John Wiley & Sons.

2. Banks, Jerry, Carson, John S. 1984. Discrete-Event System Simulation. Englewood Cliffs, NJ. Prentice-Hall.

3. Pegden, Denis C., Shannon, Robert E., Sadowski, Randall P. 1990. Introduction to Simulation Using Siman. New York, NY. McGraw-Hill, Inc.

5. Russel, Edward C., 1983. Building Simulation Models with SIMSCRIPT II.5. Los Angeles, CA. CACI, Inc.

20. Proceedings of the Winter Simulation Conference, eds. B.L. Nelson, W.D. Kelton, G. M. Clark, Phoenix, Arizona.

21. Proceedings of the Summer Simulation Conference, eds. D. Pace, Baltimore, MA.

5. Application examples

1. Schriber, Thomas J. 1991. An Introduction to Simulation Using GPPS/H. New York, NY. John Wiley & Sons.

16. Pritsker Corporation. 1989. Case Histories - A Pritsker Simulation Solution. Indianapolis, IN. Pritsker Corporation.

17. CACI. 1989. Major Application of SIMFACTORY. La Jolla, CA. CACI.

6. Advantages and disadvantages of simulation

1. Schriber, Thomas J. 1991. An Introduction to Simulation Using GPPS/H. New York, NY. John Wiley & Sons.

2. Banks, Jerry, Carson, John S. 1984. Discrete-Event System Simulation. Englewood Cliffs, NJ. Prentice-Hall.

3. Pegden, Denis C., Shannon, Robert E., Sadowski, Randall P. 1990. Introduction to Simulation Using Siman. New York, NY. McGraw-Hill, Inc.

7. Choosing simulation software

2. Banks, Jerry, Carson, John S. 1984. Discrete-Event System Simulation. Englewood Cliffs, NJ. Prentice-Hall.

18. Banks, Jerry. 1991. Selection of Simulation Software. Presented at "Time Advance Using Computer Simulation", Vienna, Austria.

19. Banks, Jerry. 1991. Selecting Simulation Software. In Proceedings of the Winter Simulation Conference, eds. B. L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

8. Current trends and future perspectives of simulation

7. Ulgen, O.M., Thomasama T., Otto, N. 1991. Reusable Models: Making Your Models More User-Friendly. In Proceedings of the Winter Simulation Conference, eds. B.L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

8. Raczynski, S. 1991. PASION Tutorial. In Proceedings of the Winter Simulation Conference, eds. B. L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

9. Bischak, D.P., Roberts, S.D. Object-Oriented Simulation. In Proceedings of the Winter Simulation Conference, eds. B.L. Nelson, W.D. Kelton, G. M. Clark, Phoenix, Arizona.

10. Rothenberg, J. Tutorial: Artificial Intelligence and Simulation. In Proceedings of the Winter Simulation Conference, eds. B. L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

11. Hurley, C. Object-Oriented Graphical Analysis. In Proceedings of the Winter Simulation Conference, eds. B.L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

12. Cohen, P.R. Integrated Interfaces for Decision-Support with Simulation. In Proceedings of the Winter Simulation Conference, eds. B.L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

13. Gordon, R. F., Loewner, P. G., MacNair E. A., Wang, H., Gordon, K. J., Kurose J. F., Error Detection and Display for Graphical Modeling Environment. In Proceedings of the Winter Simulation Conference, eds. B. L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

14. Roocks, M. A Unified Framework for Visual Interactive Simulation. In Proceedings of the Winter Simulation Conference, eds. B. L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

15. Fishwick, P. A., Porr H. A. 1991. Using Discrete Event Modeling for Effective Computer Animation Control. In Proceedings of the Winter Simulation Conference, eds. B. L. Nelson, W. D. Kelton, G. M. Clark, Phoenix, Arizona.

22. Niwinski, J. 1990. SEIMI - Simulation Based Decision Support Module for Scheduling in a Flexible Shop Floor. In Proceedings of the 23rd International Symposium on Automotive Technologie and Automation ISATA Advanced Automotive Manufacturing, Vienna, Austria.

23. Niwinski, J., Ruzicka R. 1991. On-line Simulation Using SIMUL_R. In Proceedings of the Summer Simulation Conference, eds. D. Pace, Baltimore MD.

24. Niwinski, J. 1991. A Simulation Based Decision Support System for a Flexible Job Shop. In Proceedings of the Summer Simulation Conference, eds. D. Pace, Baltimore, MD.

* * * * *

The following ten articles were given to us by the author of our main article. Those articles appeared, inter alia, in the "1991 Winter Simulation Conference Proceedings", the Arizona Biltmore, Phoenix, Arizona, 8-11 December 1991 and were edited by Barry L. Nelson, The Ohio State University, W. David Kelton, University of Minnesota and Gordon M. Clark, The Ohio State University.

2. SIMSCRIPT II.5 AND MODSIM II: A BRIEF INTRODUCTION

Dr. Edward C. Russell

Russell Software Technology
1735 Stewart Street
Santa Monica, California 90404

Abstract

The SIMSCRIPT II.5 and MODSIM II programming languages are described. SIMSCRIPT II.5 with its integrated graphical interface, SIMGRAPHICS, substantially reduces time and effort in simulation model development. Its English-like syntax improves readability of the code and substantially reduces the need for documentation. MODSIM II is a modern object-oriented language with built-in support for simulation that also has an integrated graphical interface. It is a compiled language which is available for most systems including mainframes, workstations and PCs. The built-in object-oriented constructs of MODSIM II include single and multiple inheritance, dynamic binding of objects, polymorphism, data abstraction and information hiding.

1. Introduction to SIMSCRIPT II.5

SIMSCRIPT II.5 is a well established, standardized, and widely used language with proven software support. It assists the analyst greatly in the formulation and design of simulation models and gives the programmer and analyst a common language for describing the model. The benefits of using SIMSCRIPT II.5 can be felt at all stages in the development of a model, including:

The powerful "world-view" consisting of Entities, Attributes and Sets provides a natural conceptual framework in which to relate real objects to the model.

The modern, free-form language contains structured programming constructs and all the built-in facilities needed for model development. Model components can be programmed so as to clearly reflect the organization and logic of the modelled system.

A well-designed package of program testing facilities is provided. Tools are available to detect errors in a complex computer program without resorting to memory dumps and other archaic means.

The SIMSCRIPT program structure allows the model to evolve easily and naturally from simple to detailed formulation as more information becomes available. Many modifications, such as choices of set disciplines and performance measurements are simply specified in the program preamble in a non-procedural manner. Animation and presentation graphics can even be changed without program modification.

The powerful English-like language allows for modular implementation. Because each model component is readable and self-contained, the model listing can be understood by the end-user who may not be at all familiar with programming. Because the detailed model documentation is the program listing, it is never obsolete or inaccurate.

1.1 Overview

The purpose of a simulation must be clearly articulated before embarking on model development. Many modelling efforts have been doomed to failure, because a clear goal was never determined. The natural tendency is to model in great detail that part of the system which is well understood and "sweep under the rug" or over-simplify those parts which are not understood. The detailed model of the well understood parts yields many lines of model code and gives the illusion of great progress, when in fact, a much smaller model of the entire system may actually be of much greater value. In general, a model is an abstraction of the real system under study. It is not necessary or even desirable to include all of the details of the actual system. Deciding which details are essential and which may be omitted for the purposes of the study is perhaps the most difficult task which the modeller faces.

Without its world-view, SIMSCRIPT II.5 would be just another programming language, albeit a very powerful one. But with its world-view, the modeller is guided in the formulation of a complete specification of the problem. The objects in the real world map very naturally into the SIMSCRIPT II.5 objects, which break down into classes termed TEMPORARY ENTITIES, PROCESSES, and RESOURCES. (All capitalized words are part of the SIMSCRIPT II.5 vocabulary.)

Any entity may have ATTRIBUTES which give it individual characteristic values. While all instances of a particular entity class have the same named attributes, each instance has its own values for the attributes. In addition, entities may be organized into SETS in order to represent any type of ordered list with various ordering disciplines (FIFO, LIFO, or RANKED by any combination of attribute values).

After the static structure of the model has been described, the dynamic aspects are described in terms of process routines. Each process routine corresponds to a declared process entity. Very natural commands are employed for manipulating objects in the process routines. Processes may WORK or WAIT for a period of simulated time. They may be FILEd in sets or REMOVEd from them. They may ACTIVATE, INTERRUPT or RESUME one another. Processes may REQUEST or RELINQUISH resources, automatically waiting for those which are unavailable when requested and automatically starting other processes when relinquishing unneeded resources.

Animation in SIMSCRIPT II.5 is a very natural extension of the established world-view. Entities may be declared to be GRAPHIC in order to participate in animated displays. The actual form of the display (the so-called "icon") is described through the use of an editor and may be changed independently of the model.

1.2 SIMSCRIPT II.5 language features

SIMSCRIPT II.5 is a complete programming language. In addition to its simulation modelling capabilities, it has a full range of input/output capabilities including the ability to specify either formatted or freeform input, screen-oriented output (including cursor placement), generalized reports which may expand to multi-page width as well as length. The TEXT mode of variable declaration permits very general text manipulation of character strings of arbitrary length, including operations such as concatenation, substring search and replace, case change, etc.

The entity/attribute/set structure mentioned above is an extension of a very powerful underlying data structure. Arrays in SIMSCRIPT II.5 may be of any dimension whatever, without limit. The allocation of storage for the arrays occurs during execution and arrays may be deallocated and reallocated with different dimensions.

The support of the representation of statistical phenomena is extensive. Generators exist for random numbers distributed according to uniform, integer uniform, normal, lognormal, exponential, beta, gamma, Erlang, Poisson, binomial, triangular, and Weibull distributions. If these are not sufficient, an arbitrary numerical distribution is available to describe any distribution as a table of values versus probability (individual or cumulative).

The collection of data in the form of statistical performance measures is supported by three very powerful statements: ACCUMULATE, TALLY, and COMPUTE. ACCUMULATE and TALLY update statistical counters as the variable of observation changes values. Then only when the results are needed are the final statistical calculations performed. The measures available include number of samples, sum, average, maximum, minimum, standard deviation, variance, sum of squares and mean square. ACCUMULATE performs these calculations on a time-dependent basis, while TALLY performs them on a sample basis.

Part of the ongoing development effort of SIMSCRIPT II.5 is to make the interface between user and model easier to understand. Models can be developed in which the parameters can be easily represented as presentation graphics such as pie charts, strip charts, dials level meters, bar graphs, etc. These so-called smart icons are updated on the screen as the simulation proceeds. In addition, animation capabilities have been developed to display moving objects against a static background in order to give further insight into the complex interactions which take place within a system.

The preparation of the presentation graphics as well as the icons for animation is accomplished through the use of editors. The icons are stored with the program but may be modified without having to modify the program or clutter it with non-system related code.

2. Introduction to MODSIM II

MODSIM II was specifically designed to support large programming projects. It is a compiled, modular, object-oriented language with multiple inheritance. To protect the user's investment in applications, MODSIM can be moved to new computer systems as they become available. Its syntax is based on that of Modula-2. Modularity in MODSIM II improves reliability and code reusability. Objects and routines performing related functions can be grouped into modules. These can be put into libraries for reuse by other programs. The simulation constructs are based on those used in SIMSCRIPT II.5. The portability of MODSIM II derives from the fact that its compiler emits C code which is compiled, in turn, by each computer's C compiler.

Finally, the integrated dynamic graphics of MODSIM II substantially reduces the time and effort needed to display results with animation and presentation graphics. It only takes a few statements to make dynamic icons, histograms, clocks and meters appear and change as the simulation runs. MODSIM II is a complete, general purpose programming language which is ideal for large software engineering projects.

2.1 Objects

An object is essentially an encapsulation of data and code. The data describes the object's current status. The code describes what the object does. As an example of an object in MODSIM II, consider things that move around, such as trucks and airplanes. This is the type definition of a moving object:

```
TYPE
     MovingObj =
     OBJECT
       position : LocTyp;
       course   : [ 0 .. 359 ];
       speed    : INTEGEP;
       TELL METHOD GoTo(IN dest : LocTyp,
                        IN spd  : INTEGER);
       ASK  METHOD Stop;
     END OBJECT;
```

MovingObj is an object type. It has three data fields which hold information about its location, course and speed. In addition it has two methods. Methods are an object's procedures or routines which define its behaviour. GoTo makes the object go to the specified destination from its current position. Stop is used to set the object's speed to zero. Note that the above object type declaration simply describes the state and methods of MovingObj and serves as an interface to the object. The actual code for the methods is supplied separately in the object declaration block. For example:

```
OBJECT MovingObj;
     TELL METHOD GoTo(IN dest : LocTyp,
                      IN spd  : INTEGER);
     VAR
       travelTime : REAL;
     BEGIN
       travelTime := ...  {compute time}
       course := ... {some trig calculation}
       speed  := spd;
       WAIT DURATION travelTime
          {simulation time elapses here}
       END WAIT;
       speed := 0;
       position.x := dest.x;  {update}
       position.y := dest.y;  {position}
     END METHOD;
     ASK METHOD Stop;
     BEGIN
       speed := 0;
     END METHOD;
END OBJECT;
```

ASK methods are instantaneous with respect to simulation time. When an ASK method is invoked, the caller pauses and control passes to the invoked ASK method. When the invoked method completes, the caller resumes. ASK methods behave like a procedure call but have direct access to all fields and methods of that object. No simulation time can pass in an ASK method.

TELL methods are asynchronous. When the TELL method is invoked, it is simply scheduled for execution, and the caller immediately continues execution without waiting for the TELL method to start. Simulation time can elapse in a TELL method.

A TELL method starts execution under control of the built-in simulation timing routine. The data fields of an object instance are visible to all other parts of a program and may be read using an ASK statement. However an object's fields may be changed only by the object itself. To use an object, we create an instance of that object type and send it messages using ASK or TELL when we want it to do something.

2.2 Information hiding

As we have seen, the fields of an object can be changed only by the object itself. This is one level of information hiding. However it is still normally possible for any program code to "read" the value of an object's fields using an ASK statement. We can achieve a higher level of information hiding by declaring some of the fields to be private. Private fields cannot be seen except by the object itself. Methods can be PRIVATE, too. Methods which are private can be invoked only by other methods of the object.

2.3 Inheritance

MODSIM II supports inheritance. With inheritance, new object types can be defined in terms of existing object definitions. While most languages only allow inheritance from one existing object type, MODSIM II supports multiple inheritance.

Here is a VehicleObj type definition created from a MovingObj:

```
VehicleObj = OBJECT(MovingObj)
  payload : REAL;
  TELL METHOD Load(IN amount   : REAL);
  TELL METHOD Unload(IN amount : REAL);
END OBJECT;
```

VehicleObj inherits all of the fields and methods of a MovingObj. In addition it adds a payload field and methods for loading and unloading the vehicle.

If an inherited method is no longer appropriate for the newly defined object, it can be overridden and replaced by a new one of the same name. The old method can be invoked by the replacement method as part of its behaviour if desired.

Different object types can adapt methods to fit their own particular behaviour. This important and versatile object-oriented capability is known as polymorphism - multiple behaviours invoked with the same method name.

2.4 Discrete event simulation and processes

Simulation is supported directly, as in SIMSCRIPT II.5, by built-in language constructs. The WAIT statement is used to make simulated time pass. Here is an example using the Load method of VehicleObj.

```
    TELL METHOD Load {IN amount : REAL};
CONST
    rate = 0.25; {seconds per passenger}
VAR
    loadingTime : REAL;
BEGIN
    loadingTime := amount/rate;
    WAIT DURATION loadingTime
      OUTPUT("Loading completed");
    ON INTERRUPT
      OUTPUT("Loading NOT completed");
    END WAIT;
END METHOD {Load};
```

The WAIT DURATION statement causes the method to suspend execution for the indicated amount of simulation time. Once the wait is started, control returns to the scheduler which then starts execution of the next most imminent process. When the WAIT is complete, control returns to this method at the statement after the WAIT. Any of the methods of an object which are waiting for completion can be interrupted. If the method receives an interrupt command, it executes the part of the WAIT statement after ON INTERRUPT.

Two other forms of the WAIT statement let methods synchronize themselves.

```
    WAIT FOR Flight217 TO Load(324.0);
      ...
    END WAIT;
```

This statement schedules the Load method of FLight217 but does not allow the invoking code to proceed with execution until the Load method has completed. Note that this is different from a normal TELL invocation which proceeds without waiting.

The other form of the WAIT statement uses the built-in trigger object to synchronize methods.

```
    WAIT FOR ControlTowerLight TO Fire;
      ...
    END WAIT;
```

This statement makes Flight217 wait for a signal from the ControlTowerLight before it moves. ControlTowerLight is a trigger object which has a TELL method called Trigger.

```
    TELL ControlTowerLight TO Trigger;
```

The Trigger method releases all waiting methods when it is executed.

2.5 Development environment

Transporting programs from one computer system to another has often been a problem. Frequently programs have to be extensively rewritten to eliminate machine dependencies. MODSIM II avoids this problem. It was designed for portability. MODSIM II compiles its source code to C. The MODSIM II compilation manager then compiles and links the C code to a standalone executable.

MODSIM II's compilation manager was designed to facilitate project management of large computer programs consisting of many separate modules and libraries. It manages separate compilation of MODSIM II programs consisting of multiple modules by determining which modules have been edited since the last compilation and then recompiling only those edited modules and any modules which depend on them. This process is accomplished automatically without need for "make" or project files to describe the process.

2.6 Dynamic graphics

Graphically displaying results has typically been a tedious programming task. Graphics programming is made simpler through MODSIM's interface to the SIMGRAPHICS II graphics editor and environment. SIMGRAPHICS II has three major capabilities:

- Animated graphics tied to objects in a program;
- Dynamic or static graphs tied to variables and statistics in a program;
- Interactive input menus in a contemporary windowed style.

Animated icons, graphs and input menus are all interactively edited using the SIMGRAPHICS II editor. These are then tied to existing objects and variables in the user's program. This greatly simplifies the task of creating a graphical user interface. The amount of coding for graphics is drastically reduced. An important side benefit of the editor is that the appearance of objects can be edited without recompiling or changing code. This facilitates both design and subsequent maintenance as well.

Figure 1 shows a screen from a communications satellite model. The satellites are icons which move around the earth. The line between two satellites indicates that a message is being passed. At the top left is a trace plot of message rate versus time. At top right is a level meter showing the current mean message rate. Finally, the clock at the bottom shows that we are 31 seconds into the simulation.

2.7 Benefits of MODSIM II

Any high order language is designed to reduce the effort needed to program a set of problems. The object-oriented and modular features of MODSIM II substantially reduce the time and effort needed to write programs.

- Objects improve reliability because they encapsulate data fields and provide a disciplined interface to these fields;
- Development time is reduced because code can be put in libraries and reused;
- Modules permit step-wise development, particularly by separating the definition module from the implementation module;
- Inheritance allows programmers to build on top of previous effort instead of starting from scratch each time;
- Integrated dynamic graphics substantially reduce the time and effort needed to build menus and display results.

2.8 Conclusions

MODSIM II is a robust, general-purpose programming language with built-in graphics. Its features substantially reduce the time and effort required to write and validate computer programs.

2.9 Availability

SIMSCRIPT II.5 and MODSIM II are proprietary products of CACI Products Company. They are sold on a free-trial basis. A special university program of CACI supplies SIMSCRIPT II.5 and MODSIM II to educational institutions for the cost of distribution.

References

CACI. 1988. SIMSCRIPT II.5 Programming Language, CACI Products Company, La Jolla, California.

Law, A. M. and C. S. Larmey. 1984. An Introduction to Simulation Using SIMSCRIPT II.5. CACI Products Company, La Jolla, California.

Russell, E. C. 1989. Building Simulation Models with SIMSCRIPT II.5. CACI Products Company. La Jolla, California.

Belanger, R., B. Donovan, K. Morse and D. Rockower. 1990. MODSIM II Reference Manual. CACI Products Company, La Jolla, California.

Belanger, R., and A. Mullarney. 1990. MODSIM II Tutorial, CACI Products Company. La Jolla, California.

Author biography

Dr. Edward Russell has over twenty years' experience applying simulation to commercial and government problems.

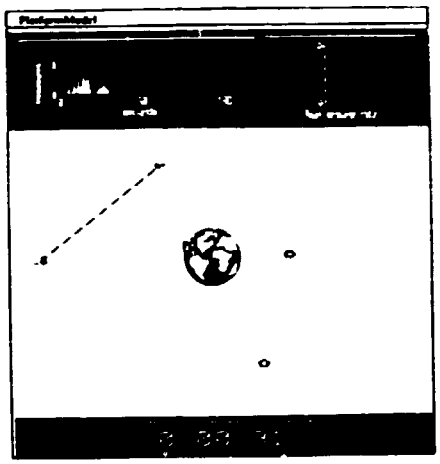


Figure 1. Satellite Communication Simulation

3. INTRODUCTION TO SIMFACTORY II.5

John Goble

CACI Products Company
3344 North Torrey Pines Court
La Jolla, California 92037

Abstract

This paper provides a brief introduction to SIMFACTORY II.5 and explains how models can be developed without programming. The type of users who benefit most from SIMFACTORY II.5, the types of systems SIMFACTORY II.5 can model, and the various implementations of SIMFACTORY II.5 are also described.

1. Introduction

SIMFACTORY II.5 is a factory simulator written in SIMSCRIPT II.5 that provides its user with the ability to quickly model factories without programming. This capability is made possible with a mouse-driven graphical user interface that enables the user to build a graphical representation of his factory. This paper describes who should use SIMFACTORY, the types of systems SIMFACTORY can model, and how a model is constructed.

2. Who should use SIMFACTORY II.5?

SIMFACTORY II.5 has been written for engineers whose other duties make it impossible for them to work on a simulation full time. Usually, this is because they have many other non-simulation tasks to perform and yet find a need for simulation in their work. In many cases these engineers will do without simulation altogether rather than use a programming language. However, there are often times when experienced simulation users require a model in less time than is possible with a language. In either case, ease of use and rapid production of working models are extremely important. That is what SIMFACTORY is designed to provide.

3. What can SIMFACTORY model?

As the name SIMFACTORY indicates, SIMFACTORY is used to model systems that act like factories. Most of the time this refers to manufacturing operations. However, many systems have been modelled with SIMFACTORY that are outside the manufacturing sector and yet have much in common with factories. For example, an insurance company wanting to know how many people were necessary to handle incoming phone calls modelled their operation with SIMFACTORY. This was possible because their operation was basically a factory that processed incoming calls. Others have used SIMFACTORY to model the flow of paperwork, viewing an office as a factory that processed paper.

4. How is a model developed?

Modelling with SIMFACTORY is most successful when it is performed in an iterative manner, starting with a fairly simple and therefore manageable representation of the factory. After the initial model is developed and working it is saved and copied. The copy is then enhanced until it reaches the next milestone in the development of the model. Again the model is saved and a copy is made for further refinement. This process is repeated until the last milestone in the model is reached.

In SIMFACTORY we call the first simplified model of the factory the basic model. A basic SIMFACTORY model represents only the stations, queues and transportation paths that exist on the factory floor. Transporters and conveyors are ignored. Even though many products may be made in the factory only two or three are included in the model at this time. Any information about shifts, equipment failures, tooling, and transporters is not entered until later. The objective is to create a working model that can be progressively refined until the desired level of detail has been reached.

The basic model is built by first defining the factory layout, then the products produced by the factory, and finally, by setting the run options indicating the run length, number of replications and so on. The complete process is explained in the following paragraphs.

5. Define the layout

The layout consists of processing stations, buffers (queues), receiving areas, and transportation paths. It is created by selecting and positioning icons that represent these components. As each icon is positioned the data that describes its characteristics (name, capacity, setup time, etc.) are entered. Of course, editing capabilities such as copying, moving, or deleting icons are available for making changes at a later time. After the icons are positioned and described, the transportation paths that connect one icon to the next are drawn. Figure 1 shows a layout of a simple gear finishing line.

6. Processing stations

Stations represent anything that processes or changes a part in some way. These could be machines such as lathes and mills or perhaps an inspection area where an inspector visually examines the parts. In SIMFACTORY stations are described in terms of the operations they perform. The part remains in the station for the amount of time called out in the Process Plans. (The plans are explained later in this article.) After processing, the part will be sent downstream for further work. However, if none of the downstream stations or queues are able to accept the part it will remain in the processing station until a station or queue becomes available. Three types of processing stations are available: Normal, Chamber and Batch.

A Normal Station may perform any number of single operations on a part and then sends the part downstream. When the Normal Station is busy it will not accept any more work until it has finished processing and unloaded the current part(s).

Chamber Stations differ from Normal Stations in that they may accept additional parts even after they have already started working on other parts. The limiting factor on a Chamber Station is its capacity, which is the maximum number of parts that may be in the Chamber at one time.

6.1 Buffers

Buffers are simply areas where parts wait before the next operation is performed. Parts may accumulate in the queue until its capacity has been reached. At that time no more parts may enter the queue until it unloads some parts.

6.2 Receiving areas

New work that is entering the factory enters in Receiving Areas. Work may be scheduled by assigning a quantity of parts to arrive periodically or by using a schedule to specify the exact time of arrival.

6.3 Transportation paths

Another component of the factory layout is the transportation paths, which indicate the path from one station to the next and which buffer(s) feeds which station(s).

7. Define the products

In SIMFACTORY the steps necessary to make a product are defined in the Process Plans in terms of the operations necessary to produce the product. Process Plans determine what operations are performed on the part, the duration of each operation, and the order in which the operations are to be performed. Assemblies, disassemblies, and branching (such as occurs at an inspection station where the part passes part of the time and fails part of the time) are all shown in the Process Plans. This approach also makes it possible to show multiple products in production on the same production line. In fact, each product may have its own unique set of processing times. Rework loops are easily constructed even if different processing times are used on the second pass through the line.

A process plan consists of three lists: a list of input parts to the plan, a list of operations performed on the parts, and a list of output parts produced by the plan. The input parts may either be raw materials or work-in-process produced by another plan. The operations list references the operations performed by the stations on the factory floor. This list of operations combined with the information on the factory floor tells SIMFACTORY how to route the parts through the factory. The output parts may be finished goods, scrap, or work-in-process.

8. Run the basic model

The last step in building the basic model is to set the length of the run, the number of runs to make, the length of the warm-up period, and the reports that will be generated by the model.

At this point the basic model is fully defined and should be run. Obviously the output is not yet what is needed for analysis and decision making, but the output should be checked to see if this version of the model seems to be working properly.

9. Define additional products

After the basic model is working the first refinement that should be made is to define additional products. If a large number of products are being modelled then they should be defined two or three at a time. Then when the model works properly with the newly defined products move on and add more.

10. Define resources

In SIMFACTORY the term resources refers to anything that is necessary to carry out an operation at a processing station. For example, a resource could be some tooling or a specially skilled operator.

Resources are added to the basic model in two steps. In the first step the resource is defined and the quantity available at the start of the run is set. In the second step each station requiring resources is defined. This is done by indicating what resource (or resources) is required by the station and the quantity required.

11. Define the transporters

SIMFACTORY transporters may either be batch transporters such as forklifts, or conveyors. To define a transporter you first position the transporter in the layout and then define the characteristics of the transporter. The important characteristics of a batch transporter are its pickup speed, delivery speed, load time, unload time, and capacity. A conveyor has a speed and capacity. In both cases the paths of the transporter and conveyors must also be defined. This is done by indicating which paths comprise the transportation zone of the transporter and conveyor.

12. Define the interruptions

Interruptions are any activity that interferes with the operation of a station or transporter. The two most common examples are equipment failures and preventive maintenance. in SIMFACTORY we would say the failure is a priority interruption because it takes priority over anything else the station could be doing. Preventive maintenance will only occur when the station is between operations.

Other characteristics of interruptions that can be specified are the mean time between interruptions, the type of interruption clock, and the mean time to resume. The mean time between interruptions is the time from one interruption to the next. This can be calculated from the end of one interruption to the end of the next, or it could be the time between interruptions, or it can be based on the calendar time, or the operating time of the station or transporter being interrupted. The mean time to resume specifies the duration of the interruption.

13. What reports are available?

During the simulation traces and snapshot reports are available to track the progress of the simulation. Traces provide detailed information about each event as it occurs in the model. This will include events such as the arrival of raw materials, the start of an interruption, the completion of a final product and so on.

Snapshots provide a picture of the model at a specific point in time.

After the simulation, reports are available that summarize how the model performed. Information on such things as equipment utilization, throughput, product makespan and queue utilization is available. Each run (or replication) is summarized in a set of reports. A summary report of all the runs provides means, standard deviations and confidence intervals on the model output.

14. How is the animation prepared?

The animation of the factory automatically follows from the description of the model. In other works, there are no extra steps to prepare the animation. However, animation can be improved by creating custom icons in the Icon Editor.

15. Implementation of SIMFACTORY II.5

SIMFACTORY is available on the following computers: IBM PC ATs and compatibles (under DOS, Microsoft Windows and OS/2), IBM PS/2s, HP 9000 Series 300 and 800, the Sun 3 and the Sun 4. The look of SIMFACTORY on each of these machines is consistent, as are the data files. Users will have little trouble working on the different implementations and moving data from one machine to the other.

16. Summary

SIMFACTORY's graphical representation of the factory together with sound modelling practices make SIMFACTORY the ideal tool for rapid model development. It should be seriously considered by anyone who is short on time but who requires a model for their analysis or presentation.

References

CACI. 1990. SIMFACTORY II.5 User's Manual, La Jolla, California: CACI Products Company.

Goble, J. 1990. SIMFACTORY Course Notes. CACI Products Company, La Jolla, California.

Russell, E. C. 1983. Building Simulation Models with SIMSCRIPT II.5, CACI Products Company, La Jolla, California.

Author biography

John Goble heads the SIMFACTORY II.5 development team. He has experience teaching and consulting with manufacturing engineers.

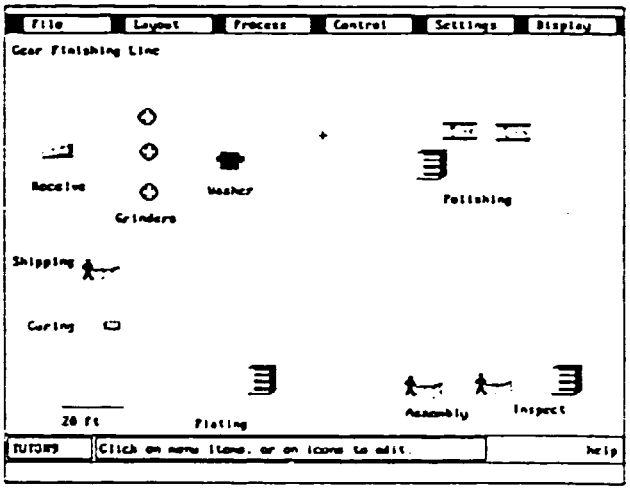


Figure 1: Layout of a Factory

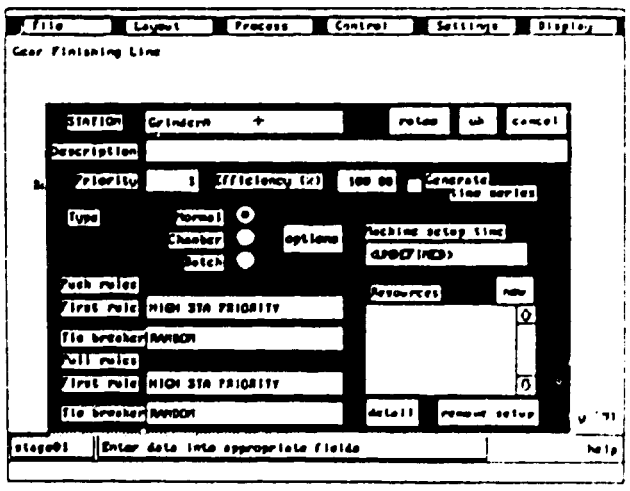


Figure 2: Editing a Station Description

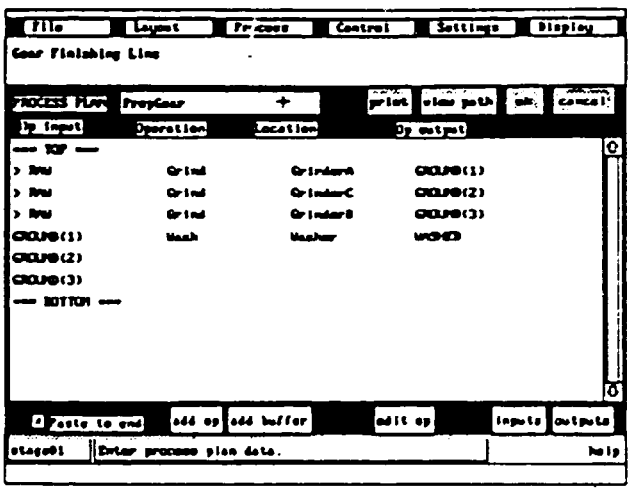


Figure 3: Editing a Process Plan

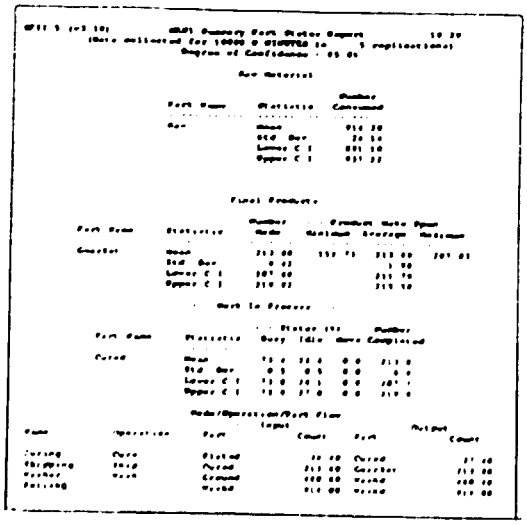


Figure 4: Summary Report from a Simulation

4. PERSPECTIVES ON SIMULATION USING GPSS

Thomas J. Schriber

Computer and Information Systems
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48109-1234

Abstract

A brief perspective on GPSS (General Purpose Simulation System) is presented. The approach taken in GPSS to model a one-line, one-server system is explained, implementation details are provided, and results are discussed. Suggestions for further study are given.

1. GPSS in brief

GPSS (General Purpose Simulation System) is a simulation modelling language used to build computer models for discrete-event simulations. (A discrete-event simulation is one in which the state of the system being simulated changes at only a discrete set of time points.) GPSS lends itself especially well to modelling systems in which discrete units of traffic compete with each other for the use of scarce resources, and is useful in determining how well such systems will respond to the demands placed on them. GPSS has been applied, for example, to the modelling of manufacturing systems, communication systems, computing systems, transportation systems, inventory systems and health-care systems, among others.

2. GPSS semantics and syntax

GPSS offers rich semantics with sparse syntax. For example, only seven statements (plus several run-control statements) are required to model a one-line, one-server queuing system in GPSS. These statements take such simple forms as "GENERATE 18,6" and "QUEUE LINE". No read, write, format, or test statements appear in the model. And yet, when a simulation is performed with the model, fixed-form, fixed-content output is produced, providing statistics for the server (e.g., number of times captured; average holding time per capture; fraction of time in use) and the waiting line (e.g., average contents; maximum contents; average time in line), etc. This limited example is roughly suggestive of the character of GPSS. A GPSS model for a one-line, one-server system is given here in Appendix A.

The sparse syntax of GPSS, coupled with its block-diagram orientation, makes it possible for the beginner to learn quickly a usable subset of the language. Because GPSS is rich and versatile, however, serious study is required to master the language.

3. Various GPSS implementations

GPSS is a multi-vendor language. (This is in contrast with such languages as SIMAN, SLAM, and SIMSCRIPT.) Brief comments are provided here on several GPSS vendors and their GPSS implementations.

Wolverine Software Corporation (4115 Annandale Road, Annandale VA 22003-2500, telephone 800-456-5671 or 703-750-3910) vends GPSS/H, Release 2, which runs on a wide range of hardware platforms from DOS machines through engineering workstations to mainframes. In addition to the vendor's GPSS/H reference manual [Henriksen and Crain 1989], two textbooks describe this implementation in introductory fashion [Banks, Carson and Sy 1989, and Schriber 1991] and come with Student DOS GPSS/H on an included disk. The vendor sponsors GPSS/H courses and courses teaching Proof, the vendor's high quality and inexpensive presentation and animation software.

MINUTEMAN Software (P.O. Box 171, Stow, MA 01775-0171, telephone 800-223-1430 or 508-897-5662) vends a series of GPSS/PC implementations. GPSS/PC provides built-in graphics and animation and in some versions interfaces with AUTOCAD. The vendor supplies a reference manual [Cox 1988] and a series of tutorial models [Cummings 1988]. A textbook containing Student GPSS/PC on an included disk is Karian and Dudewicz [1991]. The vendor has information about courses that train participants in the use of GPSS/PC. Contact the vendor for information about courses, books, and reference materials.

Simulation Software Limited (760 Headly Drive, London, Ontario, Canada N6H 3V8, telephone 519-657-8229) vends a series of GPSS implementations. Contact the vendor for details.

International Business Machines (IBM) leases GPSS V, its early 1970's implementation of GPSS which, although out of date, is still used occasionally. Contact your IBM representative for details.

4. The GPSS tutorial

In the GPSS tutorial at the Winter Simulation Conference, the fundamentals of queuing system logic and the elements offered by GPSS to implement this logic will be discussed. A GPSS model for a one-line, one-server queuing system is given below in Appendix A for interested persons unable to attend the tutorial.

Appendix A: A GPSS model for a one-line, one-server system

This appendix presents a GPSS model for a one-line, one-server queuing system. Although the model is largely generic to GPSS, its implementation is shown in GPSS/H. Animation of the model is not discussed but could be accomplished in MINUTEMAN Software's GPSS/PC or by using Wolverine Software's animation product, Proof. Contact the vendors for details.

The appendix consists of these sections:

A.1 Statement of the problem;
A.2 The approach taken in building the model;
A.3 The GPSS block diagram for the model;
A.4 The GPSS model file;
A.5 Selected simulation output;
A.6 Suggestions for further study.

A.1 Statement of the problem

In a manufacturing system, castings are sent to a drill, where each casting is to have a hole drilled in it. The interarrival time of castings at the drill is uniformly distributed over the interval 15.0 ± 4.5 minutes. The time required to drill a hole in a casting is 13.5 ± 3.0 minutes, uniformly distributed. Castings are processed in first-come, first-served order. Model this system in GPSS, making provision to collect queuing statistics for castings waiting their turn to be drilled. Perform a single simulation with the model, simulating until holes have been drilled in 100 castings. Briefly discuss the output produced at the end of the simulation.

A.2 Approach taken in building the model

Consider the series of events experienced by a casting as it moves through the one-line, one-server system:

1. The casting arrives at the system.
2. The casting requests the drill.
3. The casting waits, if necessary, to capture the drill.
4. The casting captures the drill.
5. The casting holds the drill in a state of capture while a hole is drilled in the casting.
6. The casting gives up control of the drill.
7. The casting leaves the system.

Castings can be thought of as units of traffic (objects) that move through the castings-and-drill system. These units of traffic are conveniently simulated in GPSS by language elements known as "transactions". Transactions are units of traffic which are created and introduced into a model from time to time, move along a path in the model as the simulation proceeds, and then are destroyed (leave the model). The experiences of transactions as they go through their life cycle in the castings-and-drill model are analogous to the experiences of castings as they go through the castings-and-drill system. Positioned on the path along which transactions move are blocks. Movement of a transaction into a block causes the block to be executed. By choosing appropriate types of blocks, the GPSS modeller can easily build an appropriate path (sequence of blocks) for casting-transactions to move along to mimic the sequence of events outlined above.

The sequence of blocks begins with the type of block used to create transactions from time to time during a simulation and introduce them into a model, the GENERATE block. The time that elapses between introduction of consecutive transactions into a model by a GENERATE block is "interarrival time". In this model, the interarrival time is uniformly distributed over the interval 15.0 ± 4.5 minutes. (15.0 ± 4.5 describes the open interval ranging from 10.5 to 19.5.) The values 15.0 and 4.5 are provided in the model as GENERATE block operands. (In general, arbitrary interarrival-time distributions can be modelled at GENERATE blocks. This is done by using built-in or user-defined functions that describe the distribution, then specifying these functions as GENERATE-block operands. See Schriber [1991], chapter 13, for particulars.)

The sequence of blocks ends with a TERMINATE block. When a transaction executes a TERMINATE block, the block destroys the transaction. A counter can be used with TERMINATE blocks so that, after a specified destroy count has been reached (a count of 100 in this problem), a simulation will stop. (More generally, arbitrarily complicated stopping conditions can be specified in GPSS models.)

A SEIZE block is included in the sequence. A transaction requests control of a single server by trying to execute a SEIZE block. A SEIZE block operand is used to identify the single server. If the server is idle when a transaction requests it, the requesting transaction executes the SEIZE without delay and takes control of the server. But if the server is currently under the control of one transaction when another requests it, the requesting transaction cannot execute the SEIZE block. Instead, it remains in its current block and waits its turn to capture the server. In the simplest case, turns come in the order of first-come, first-served. (In general, arbitrarily complicated service orders can be specified in GPSS.)

A RELEASE block is also included in the sequence. A transaction which is in control of a single server gives up control by executing a RELEASE block. A RELEASE block operand is used to identify the server involved.

GPSS automatically collects (and then, when a simulation stops, prints out) statistical information about single servers modelled with use of SEIZE and RELEASE blocks. (See section A.5.)

An ADVANCE block is used to delay movement of a transaction along its path for a specified simulated time. In this model, an ADVANCE block can be used to simulate the time required for the machine to drill a hole in a casting ("service time"). The service time in this model is uniformly distributed over the open interval 13.5 ± 3.0 simulated minutes. The values 13.5 and 3.0 are provided in the model as ADVANCE block operands. (Arbitrarily complicated service time distributions can be modelled at ADVANCE blocks. This is done by using built-in or user-defined functions which describe the applicable distribution.) By placing an ADVANCE on the path between SEIZE and RELEASE, simulated time delays between server capture and release can be modelled.

By executing a QUEUE block, a transaction initiates membership for itself in a queue, or waiting line. This membership continues until the transaction brings its queue membership to an end by executing a DEPART block. An operand is used at the QUEUE and DEPART blocks to indicate the particular queue involved. By placing a SEIZE between QUEUE and DEPART blocks, transactions will be members of a queue while waiting their turn to capture a server. GPSS automatically collects and then prints out statistical information about such queues. (See section A.5.)

Seven types of blocks have been commented on in this section (GENERATE; TERMINATE; SEIZE;

RELEASE; ADVANCE; QUEUE; DEPART). In total, there are more than fifty types of blocks in GPSS. By appropriate use of these block types, models of complex systems can be built with considerable ease.

A.3 The GPSS block diagram for the model

The model described above is shown in the form of a block diagram in Figure A-1. The block diagram consists of a sequence of seven Blocks. (Each block type in Figure A-1 has its own unique, arbitrary geometry.)

The text appearing adjacent to the blocks in Figure A-1 (e.g., "castings arrive"; "check into the drill queue") is not part of the model, but is simply commentary which has been (optionally) provided as documentation.

A.4 The GPSS model file

Figure A-1 shows the block diagram for a GPSS one-line, one-server model. To perform a simulation with this model, the statement version of the Figure A-1 block diagram must be prepared and then supplemented with additional types of statements used to control compilation and execution of GPSS models. The resulting collection of statements must then be arranged in a model file. A model file is simply a computer file which can be used as the basis for performing one (or more) simulations.

Figure A-2 shows a model file corresponding to the Figure A-1 block diagram. The Figure A-2 model file has been supplemented for discussion purposes here at the top with a row of column labels ("STMT #", "Label", "Operation", "Operands", and "Comments") and at the left with a column of statement numbers (1, 2, 3, ..., 22).

Statements 8 through 14 correspond to the Blocks in Figure A-1. Each of these block statements consists potentially of a "Label" (no Labels are used in Figure A-2), an "Operation", and zero or more "Operands", and can (optionally) include appended documentation text ("Comments"). For example, STMT #8 corresponds to the Figure A-1 "GENERATE 15.0,4.5" block (where the "Operation" is GENERATE and the "Operands" are 15.0 and 4.5) and carries the optional appended comment "castings arrive".

Statements 2, 20 and 22 in Figure A-2 are examples of statements used to control the compilation and execution of GPSS models. They have been specified in Figure A-2 in such a way that when the model file is executed, only one simulation will take place. The simulation will stop when the 100th casting has been drilled.

Each model-file statement beginning with an asterisk (*) is a comments statement. In Figure A-2, STMT #'s 2, 3 through 7, 15 through 19, and 21 are examples of such statements.

A.5 Selected simulation output

Selected output automatically produced at the end of the simulation when the Figure A-2 model file was submitted for execution is displayed in Figure A-3. The output in Figure A-3 consists of: (a) clock values; (b) server statistics; and (c) queue statistics. Portions of this output are discussed below. (For a full discussion of similar output, see Banks, Carson and Sy 1989, Henriksen and Crain 1989, or Schriber 1991.)

(a) Clock values

As indicated in Figure A-3(a), GPSS maintains two simulated clocks: a RELATIVE CLOCK; and an ABSOLUTE CLOCK. Both clocks show that it took 1488.9+ simulated minutes to drill holes in 100 castings. (Limited space makes it impossible to explain the difference between the two types of clocks here.)

(b) Server statistics

Figure A-3(c) shows server (drill) statistics accumulated during the simulation. Several columns in the figure have been numbered here to make it easy here to refer to the information they contain. The meaning of the information in the several numbered columns will now be indicated by column number.

(1) The FACILITY column lists the identifier used in the model for the single server (the DRILL, in this case) for which statistics are being reported. (In GPSS, the facility entity is used to model single servers.)

(2) The - AVG-UTIL-DURING - TOTAL TIME column shows the fraction of total simulated time that the server was captured. In this case, the DRILL was in use 91.7 per cent of the time.

(3) The ENTRIES column indicates the number of times the server was put into a state of capture during the simulation. This statistic is a capture count. In Figure A-3(c), the capture count is 100.

(4) The AVERAGE TIME/XACT column shows the average holding time per capture of the server.

(c) Queue statistics

Figure A-3(d) shows queue (waiting-line) statistics accumulated during the simulation. Several columns in the figure have been numbered here to make it easy here to refer to the information they contain. The meaning of the information in the several numbered columns will now be indicated by column number.

(1) The QUEUE column lists the identifier used in the model for the queue (the DRILLQUE, in this case) for which statistics are being reported.

(2) The MAXIMUM CONTENTS column indicates the maximum length of the waiting line (this statistic has the value 2 in the case of the DRILLQUE).

(3) The AVERAGE CONTENTS column shows the average length of the waiting line (0.215 in the case of the DRILLQUE).

(4) The TOTAL ENTRIES column shows the count of the number of times transactions joined the waiting line (101 in the case of the DRILLQUE).

(5) The ZERO ENTRIES column shows the number of transactions which passed through the waiting line in zero simulated time.

(6) The AVERAGE TIME/UNIT column shows how much time transactions spent resident in the waiting line on average (3.172 in the case of the DRILLQUE). (Here, the term "UNIT" in the AVERAGE TIME/UNIT label means "transaction".)

A.6 Suggestions for further study

The preceeding material provides a glimpse at the particulars of discrete-event simulation using GPSS. Those interested in further exploration can do the following:

1. Contact the vendors and obtain specific information about the various implementations.

2. When attending a conference (such as the Winter Simulation Conference) at which simulation vendors have booths in the exhibition area, talk with the vendors and look at vendor demonstrations.

3. Obtain and read a textbook for the implementation(s) of interest and, if the textbook(s) come with student software and sample models on an included disk, experiment directly with the software.

4. Read GPSS application papers in areas of interest.

5. Take an intensive GPSS short course.

References

Banks, J., J. S. Carson, and J. Sy. 1989. Getting started with GPSS/H (with Student DOS GPSS/H on an included disk). Wolverine Software, Annandale, Virginia.

Bobillier, P. A., B. C. Kahan, and A. R. Probst. 1976. Simulation with GPSS and GPSS/V. Prentice-Hall, Englewood Cliffs, New Jersey.

Cox, S. 1988. GPSS/PC User's manual. MINUTEMAN Software, Stow, Massachusetts.

Cummings, G. F. 1988. GPSS/PC simulation tutorials. MINUTEMAN Software, Stow, Massachusetts.

Gordon, G. 1975. The application of GPSS V to discrete systems simulation. Prentice-Hall, Englewood Cliffs, New Jersey.

Henriksen, J. O. and R. C. Crain. 1989. GPSS/H reference manual, Third edition. Wolverine Software Corporation, Annandale, Virginia.

International Business Machines. 1970. GPSS V User's manual (SH20-0851). IBM Inc., Armonk, New York.

Karian, Z. A., and E. J. Dudewicz. 1991. Modern statistical, systems GPSS simulation: the first course. W.H. Freeman, New York.

Schriber, T. J. 1974. Simulation using GPSS. John Wiley & Sons, New York. (As of 1990, this book is published by Krieger Publishing, Melbourne Florida.)

Schriber, T. J. 1991. An introduction to simulation using GPSS/H (with Student DOS GPSS/H on an included disk). John Wiley & Sons, New York.

Author biography

Thomas J. Schriber is professor and Chairman of Computer and Information Systems in the Graduate School of Business at the University of Michigan. He teaches, does research, and consults in the area of discrete-event simulation. A member of Who's Who in America, he has authored or co-authored several dozen articles, has authored or edited eleven books, including An introduction to simulation using GPSS/H (Wiley, 1991), and regularly teaches intensive courses on GPSS-based simulation. From 1977 to 1986 he was the ACM member of the Board of Directors of the Winter Simulation Conferences, serving as Board Chairman two years. His professional affiliations include ACM, DSI, ORSA, SCS, and TIMS.

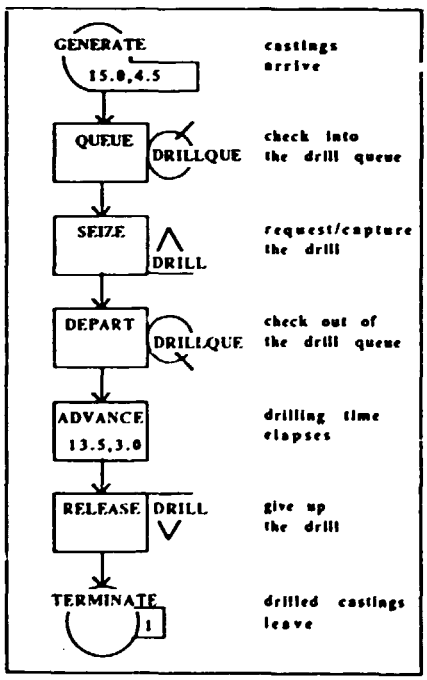


Figure A-1: GPSS Block Diagram for a One-Line, One-Server Queuing System

```
STMT #  Label    Operation  Operands     Comments

     1  ***************** One-Line, One-Server Model, WSC '91 ***************
     2           SIMULATE
     3  *                                                  Base Time Unit: 1 Minute
     4  ********************************************************************************
     5  *          Model Segment 1 (Movement of Castings Through the System   *
     6  ********************************************************************************
     7  *
     8           GENERATE   15.0,4.5     castings arrive
     9           QUEUE      DRILLQUE     check into the drill queue
    10           SEIZE      DRILL        request/capture the drill
    11           DEPART     DRILLQUE     check out of the drill queue
    12           ADVANCE    13.5,3.0     drilling time elapses
    13           RELEASE    DRILL        give up the drill
    14           TERMINATE  1            drilled casting leaves
    15  *
    16  ********************************************************************************
    17  *          Run-Control Statements                                            *
    18  ********************************************************************************
    19  *
    20           START      100          start the simulation;  proceed until
    21  *                                 100 drilled castings have left
    22           END                     end of Model-File execution
```

Figure A-2: A GPSS Model File for the Figure A-1 Block Diagram

```
RELATIVE CLOCK:    1488.9629      ABSOLUTE CLOCK:    1488.9629
(a) Clock Values

     (1)     (2)                      (3)        (4)
           --AVG-UTIL-DURING--
FACILITY   TOTAL   AVAIL   UNAVL   ENTRIES    AVERAGE    CURRENT   PERCENT   SEIZING
           TIME    TIME    TIME               TIME/XACT  STATUS    AVAIL     XACT
   DRILL   .917                        100      13.655   AVAIL     100.0
(b) Drill Statistics

     (1)      (2)       (3)       (4)       (5)              (6)
   QUEUE    MAXIMUM   AVERAGE   TOTAL     ZERO    PERCENT   AVERAGE
            CONTENTS  CONTENTS  ENTRIES   ENTRIES  ZEROS     TIME/UNIT
DRILLQUE          2      .215       101        42     41.6       3.172
(c) Queue Statistics
```

Figure A 3: Selected Simulation Output

5. PROOF ANIMATION: THE GENERAL PURPOSE ANIMATOR

Daniel T. Brunner
Nancy J. Earle
James O. Henriksen

Wolverine Software Corporation
4115 Annandale Road
Annandale, Virginia 22003

Abstract

Proof Animation™ is high quality PC-based animation and presentation software. Proof Animation runs on any 286 or better computer with a math coprocessor, EGA or VGA graphics, and a mouse. Among Proof Animation's many advanced features are smooth motion, true zooming, and presentation portability. An open architecture makes Proof Animation compatible with most popular simulation software.

1. Introduction

During all phases of a simulation project, animation is a critical tool in presenting simulation results to management and clients. Animation is also important during model building. Users of simulation software find that complex logic problems often (though not always) come to light quickly when animated. Furthermore, animation can play an important part in the overall system design process as well as in presentation and model building. In this paper we describe characteristics of Proof Animation that help simulation modellers address all of these needs.

Throughout this paper, we use the Proof Animation Release 1.0 designation wherever it is necessary to distinguish the current product from possible future versions.

2. About Proof Animation

2.1 Overview

Proof Animation Release 1.0 is a PC-based product. The minimum hardware is a 286 machine with a math coprocessor and either an EGA or VGA display. This common hardware allows Proof Animation to be portable, which is especially advantageous when projects are presented off-site.

Proof Animation Release 1.0 is a "post-processing" animation package. In order for an animation to run, two files must exist: the layout file and the trace file. Typically, a simulation model produces the trace file that drives the animation. There are trade-offs when this approach is compared to "concurrent" animation, which animates while the simulation executes - but many of the trade-offs favour post-processing.

Proof Animation offers a special "presentation mode". Professional-looking static slides can be created and interwoven with animation segments and shown on the computer screen or with a computer projection display. It may soon be possible for a Proof Animation user to take such a presentation and create a diskette that can be given to others for presentation viewing.

Proof Animation has an open architecture. This makes Proof Animation compatible with many popular simulation software packages. In fact, simulation-specific software is not necessarily needed to drive the animation. If the software used has file I/O that is capable of producing an ASCII file, it can produce a Proof Animation trace file.

Proof Animation has a geometry-based (CAD-like) internal data structure. Changing the orientation or scale of an object, static or moving, will not affect the quality of its appearance on screen. Many options are available for zooming, panning, rotating, or re-orienting the animation. Individual objects can rotate while they move around curves.

Although Proof Animation is not a full-features CAD program, it does have a mouse-driven, CAD-like drawing mode. Through a series of pull-down menus, the layout geometry and path and shape definitions can be created using the mouse. It is also possible to read existing CAD layouts into a Proof Animation layout. Proof Animation's drawing mode has been developed to easily handle a wide variety of systems. This means that diverse applications, such as network communications and health care systems, can be handled as easily as complex material handling systems.

2.2 Hardware requirements

Proof Animation Release 1.0 runs on IBM or compatible PCs, primarily because of the large installed base of colour-capable systems. The MS-DOS operating system was chosen for its large installed base and also because it behaves as a single-tasking environment (even under some third-party multitasking software). When Proof Animation can take total control of the CPU, the result is an animation with very smooth motion.

New simulation packages have been developed, such as GPSS/H 386, that take advantage of a 386- or 486-based machine's faster 32-bit architecture and large memory address space while still running under MS-DOS. This means that an MS-DOS PC can handle large simulation models. Using Proof Animation, the user can do almost any simulation and the animation on one machine.

Proof Animation Release 1.0 supports both EGA and VGA colour graphics displays at a resolution of 640 x 350. In Proof Animation, one pixel needs four bits of video memory, and four times 640 times 350 (the screen dimensions, in pixels) is 896,000 bits, or 112K bytes. Given the 256K of video memory found on nearly all EGA cards and on every VGA adapter, Proof Animation can double buffer a 112K screen - that is, improve the appearance of the animation by keeping two copies of the screen in video memory, displaying one while updating the other. Double buffering is very important for quality animation.

We are carefully studying the evolving XGA, 8514/A, TIGA, and "Super VGA" display options, all of which offer resolution at or above 1,024 by 768. Once the PC graphics market has moved

clearly in one of these directions, the resolution of Proof Animation will be enhanced.

2.3 Post-processing versus concurrent

Proof Animation Release 1.0 is post-processing, meaning the simulation runs first, then the results drive the animation. Some animation packages run concurrently, displaying state changes while the simulation runs. What are the trade-offs?

Although it is possible with most concurrently running simulation/animation software to make certain limited types of changes to the system and watch the impact, many types of changes (such as scheduling algorithm changes) are difficult or impossible to animate without advance work by the modeller.

Concurrent animation is completely dependent on the execution of the simulation model. This can be very tedious when the software is under consistent use (e.g. during the model building phase), especially if the underlying simulation software is not particularly fast at executing.

With a post-processing animator, it is possible to fast forward quickly to any point in simulated time. Concurrent animators can only do this as fast as the simulation will run.

Post-processing adds to the portability of Proof Animation. The user needs to take only the Proof Animation software to a remote location, and the target machine need not be equipped to handle the simulation software. Thanks to ongoing "demo maker" development efforts, it may soon be possible to detach animations for distribution to others who do not have the main Proof Animation program at all.

Despite all of these arguments in favour of post-processing, there are those who do not believe they can do without concurrent animation. To address this requirement, we are planning a future version of Proof Animation that is capable of concurrent animation.

3. Proof Animation features

3.1 Presentation mode

Proof Animation has a full-featured presentation mode. This lets users create "slides" made up of words, objects, screen snapshots, or even dynamic animated segments. These slides can be linked together to produce a polished, complete presentation. Awkward transitions from overhead transparencies to a computer display during a presentation are eliminated. The presentation developer can choose to highlight areas of interest (in space or time) within the animation, and thus draw the viewer's attention to particular aspects of the simulation.

3.2 Open architecture for trace events

Some animation software is integrated into a simulation language or package. In order to use the animation, the user must go through the process of building a simulation model using the integrated tool. Other animators use post-processing, but the specifications of the trace file are generally not available to the user.

Proof Animation has an open architecture. The specifications for generating trace events are public and easily followed. The most dramatic impact of Proof Animation's open architecture has initially been the quick adoption of Proof Animation as the animation engine of choice by many people using simulation software other than Wolverine Software's own GPSSH.

It is also possible to build graphical depictions of systems that have not been simulated, or to build a new simulation/animation package around the Proof Animation graphics engine. Proof Animation can also be used by non-simulationists as presentation software. The open architecture opens some wide doors for Proof Animation and its users.

The trace event architecture of Proof Animation consists of a very simple, record-oriented animation language with English-like commands. A typical Release 1.0 animation consists of a layout file and an animation trace file. The layout file contains static geometry information and definition commands. The trace file is used for recording the time-dependent information that controls all animation activity. Both files are populated with printable ASCII characters.

Here are a few examples of the easy-to-use trace event commands:

```
SET...COLOUR...
MOVE
PLACE..ON..AT..
CREATE
TIME
DESTROY
```

Normally, a small set of commands used over and over comprise the animation trace file. The process of actually writing the trace file is automated. For simulation, this means that the model writes commands such as SET COLOUR into the file each time an entity passes a certain point in the simulation. For non-simulation applications such as control algorithm debugging, the process can be similarly automated.

3.3 Geometry-based graphics data structure

A CAD-like, vector-based structure is used in Proof Animation, allowing the software to rotate an object, pan, and zoom in or out without losing the integrity of the object. In contrast, zooming in with a pixel-based system makes the object's edges appear jagged.

The power of a CAD-like data structure provides benefits in two areas. The first is the versatility of the available drawing tools. The second is the flexibility with which the display can be manipulated. Proof Animation's CAD-like architecture allows total control of the viewing environment. This is unprecedented among PC simulation animators. The geometric data structure allows complete panning, zooming, rotating, and changes of viewpoint.

Complementing the graphics data structure is a CAD-like drawing mode for creating the layout file. Using a series of menus and a mouse, the static layout, dynamic objects, and paths can be created.

3.4 CAD interoperability for design

Proof Animation is the first animation software that enables a two-way CAD interface via the DXF file format. A separate utility makes it possible to read in an existing DXF file to create

the background portion of the Proof Animation layout file. This saves time in developing the animation. Then, if changes are made in the animation layout, the utility can produce an updated DXF file (comprised only of the subset of CAD primitives available in Proof Animation). The project design team can now rely on the simulation and animation as a timely and dependable design check. If changes are needed, they can first be tested with the simulation. Once a final design is achieved, the updated animation will produce the final layout in a file that can then be read into nearly any PC CAD system.

3.5 Smooth motion

Smooth motion was a primary design goal for Proof Animation, and it has been achieved with stunning success. In most media, it is necessary to create and recreate static images rapidly in order to create the illusion of motion. This is, of course, the principle behind motion pictures and television as well as cartoon animation.

In the case of a computer and raster-based CRT screen, or the equivalent raster-based video game, the image is created as a set of discrete pixels represented in video memory. For these applications, the pixel representation must be either recreated continuously at different locations, or saved and "blitted" to different locations on the screen. This process must be repeated many times per second, or the motion will appear jerky.

How fast is fast enough? Motion pictures run at about 20 frames per second, and standard television at about 30 frames. Simulation animation software available in the 1980s was plagued by slow frame rates. Due to the discreteness of the pixels and the resulting high bandwidth images, computer displays of artificially created objects can require even higher frame rates than television, or the motion will appear to buzz or jerk. The frame rates on much of the available software have been on the order of 10 frames per second or less, while Proof Animation has starting rates of 60 to 70 frames per second.

When Proof Animation cannot move pixels quickly enough to keep up with such high frame rate, the frame rate is reduced in order to maintain a constant (though user-adjustable) ratio of animated time to "wall clock" time. With other animation software, the apparent speed of objects moving across the screen generally diminishes in such circumstances. Proof Animation performs this adjustment continuously. With Proof Animation's high starting frame rates, the effect of reducing the frame rate remains visually acceptable.

4. The layout file

A Proof Animation layout file contains basic geometry and also definition information. The basic geometry is simply the lines, arcs, fill points, and text that make up the static background of the animation. The definition information consists primarily of Object Class definitions, Path definitions, Object initializations, Message definitions, and Named View definitions.

A Proof Animation layout file is generally developed or completed using the mouse and saved from Proof Animation. The format is ASCII and open, which allows programmers to develop other software which can read and possibly modify or create the contents of a Layout file. However, most users should never need to look at the ASCII contents of a Layout file.

4.1 Object classes and objects

Among the most important constructs in Proof Animation are the Object Class and the Object.

An Object Class is a geometric description of some type of object, such as an automobile. A traffic model might have five different Object Classes: Automobiles, Trucks, Buses, Campers, and Motorcycles. In addition to shape information, an Object Class contains a few other properties such as physical clearances, colour, and an optional speed.

Although Proof Animation does not purport to implement a true "object oriented" framework, it is meaningful to call an Object an "instance" of an Object Class. Expanding on the traffic model mentioned above, one could have northbound and southbound cars; cars making continuous turning movements; red, green, or beige cars; large cars and small cars. Each of these cars is an Object, based on the single geometric description of an automobile. There can be an arbitrary number of "Automobile" Objects in the system at once, but there need be only one "Automobile" Object Class.

All of the motion and colour-changing primitives in Proof Animation operate on Objects. Most layouts are drawn directly on the screen, and the background geometry components cannot move or change colour. However, if movement or colour change is desired, then the appropriate components can be made into Objects.

4.2 Paths

The simple things the user can do with an Object include: CREATE or DESTROY, PLACE (making it visible), SET COLOUR and SET SPEED, and MOVE (causing the Object to move smoothly from points A to B). Object movement can also be achieved via a Path. A Path is a graphically defined data structure composed of references to parts of existing lines and/or arcs. Once defined, Paths are saved as part of the layout file.

The more complicated things one can do with an Object involve Paths. Actually, using Paths is very simple, because Proof Animation does all the work. The most commonly used Path command is PLACE [object] ON [Path]. Once an Object is placed on a Path, it will follow that Path until it visually comes to rest at the end of the Path (or until it is placed elsewhere or destroyed). Paths provide outstanding power in response to a single trace event command.

A variant is the Accumulating Path, which offers even more power. On an Accumulating Path, Proof Animation reflects physical reality by allowing objects to queue visually if there is a blockage. This often makes a simulation model of the system much simpler to construct, because such queueing need not always be explicitly represented in the model. A suprising number of systems behave in this manner, from certain types of conveyors to supermarket checkout lanes. Paths play an especially important part in transportation and material handling animations.

4.3 Static objects

Objects that represent moving entities are usually not permanent. Such Objects are typically

created in the trace event file. However, other Objects might persist throughout the animation. Their initial placement might be at a coordinate location rather than on a named Path. We call such an Object a Static Object. The layout file can save individual Objects that are created and placed using the mouse prior to running the animation.

The Static Objects feature was added to Proof Animation just before final release. It serves two important functions. First, this feature largely relieves the simulation model of needing to deal with the coordinates of resources depicted in the animation. Second, Static Objects allow Proof Animation to be used as a sort of model builder. If the relationship between specific Object Classes and corresponding simulation constructs can be well defined, then a user could simply place Static Objects on the screen and save the layout for further processing by a model generator program.

We must emphasize that Proof Animation, Release 1.0, is not a model builder. The main limitation is the inability to set up custom-defined parameters that a model-building user could modify when placing the Static Objects. However, the Static Objects approach offers possibilities for future enhancements.

4.4 Messages

A Message in Proof is a definitional construct. It looks and acts similar to regular text, but the contents of the text can be changed from the trace file. Messages are named, placed, and saved during layout construction. Properties such as character size, location, and rotational orientation are saved. Messages are used for later display of statistics or status information.

4.5 Named views

Proof Animation supports Named Views. At any time during layout construction or with an animation running, a user can preserve the current View by attaching a name to it. A View in Proof Animation contains properties such as centre point and scale factor. Once meaningful Named Views have been defined, a view can be accessed quickly from Draw Mode or Run Mode.

Proof Animation provides three pre-defined Views: Home, Class, and Previous. The Home view is used at start-up. The Class view is used during Class definition only. The Previous view always returns to the previous view. The Home and Class views can be modified and resaved.

Named Views would be unimportant in an animator that offered only one or a few screens' worth of animation. However, Proof Animations can be very large, and Named Views can be very helpful for navigating them. Example Named Views in a particular animation might include loading Area, Stat Summary, and Work Cell 12A.

5. Summary

Animation is a powerful addition to any simulation effort. An animation benefits the modeller in verification, validation, and presentation of results, and helps with the overall system design process.

Simulation and animation technology is improving. Wolverine Software Corporation is contributing to this improvement by providing an innovative animation package called Proof Animation. This general purpose animator boasts many important features. Among these features are the ability to create presentations, an open architecture (for compability with a variety of software), a CAD-like structure, smooth motion, and powerful drawing features.

References

Brunner, D. T. and N. J. Earle. 1991. Using Proof Animation. Annandale, Virginia: Wolverine Software Corporation.

Brunner, D. T. and N. J. Earle. 1991. Proof Animation CAD translator user's guide. Annandale, Virginia: Wolverine Software Corporation.

Brunner, D. T. and J. O. Henriksen. 1989. A General Purpose Animator. In Proceedings of the 1989 Winter Simulation Conference, eds. E. A. MacNair, K. J. Musselman, and P. Heidelberger, 249-253. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Earle, N. J., D. T. Brunner and J. O. Henriksen. 1990. Proof: The General Purpose Animator. In Proceedings of the 1990 Winter Simulation Conference, eds. O. Balci, R. P. Sadowski, and R. Nance, 106-108. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Author biographies

Daniel T. Brunner received a B.S. in Electrical Engineering from Purdue University in 1980, and an M.B.A. from the University of Michigan in 1986. He has been with Wolverine since 1986, where his responsibilities include product marketing, product development support, training, and simulation consulting. Mr. Brunner served as Publicity Chair for the 1988 Winter Simulation Conference and is Business Chair for the 1992 Conference.

Nancy J. Earle received B.S. (1982) and M.S. (1984) degrees in Industrial Engineering from Purdue University, where her concentration was in simulation. She joined Wolverine as an industrial engineer in 1989. Her responsibilities include consulting, training, technical support, and product development support. Previously, she worked for Corning Incorporated as a simulation analyst in manufacturing. While there, she developed and taught short courses in simulation. Ms. Earle is a member of SCS and will serve as Exhibits Chair for the 1992 Winter Simulation Conference.

James O. Henriksen is the president of Wolverine Software Corporation. He is a frequent contributor to the literature on simulation and has presented many papers at the Winter Simulation Conference. Mr. Henriksen served as the Business Chairman of the 1981 Winter Simulation Conference and as the General Chairman of the 1986 Winter Simulation Conference. He has also served on the Board of Directors of the conference as the ACM representative.

6. INTRODUCTION TO SIMAN IV

Cynthia J. Kasales
David T. Sturrock

Systems Modeling Corporation
504 Beaver Street
Sewickley, Pennsylvania 15143

Abstract

The SIMAN IV environment integrates the model building, running, animation and data analysis. This paper discusses the SIMAN IV simulation environment and the concepts and methods for simulating manufacturing systems using the SIMAN IV simulation language.

1. Overview of SIMAN IV

SIMAN IV is a general purpose SIMulation ANalysis program for modelling in any of three distinct orientations. For discrete change systems, either a process or event orientation can be used to describe the model. Continuous change systems are modelled with algebraic, difference, or differential equations. A combination of these orientations can be used to model combined discrete-continuous models. The remainder of this paper discusses only the process orientation.

The SIMAN IV modelling framework is based on the systems theoretic concepts developed by Zeigler (1976), which stress the distinction between the system model and the experimental frame. The system model defines the static and dynamic characteristics of the system such as machines, storage points, work pieces, etc., and their interrelationships. The experimental frame defines the experimental conditions under which the model is run. This includes such elements as machine capacities and types of statistics to be recorded. Since the experimental conditions are specified external to the model description, they may easily be changed without affecting the basic model definition. Many different "what-if" questions can be answered merely by changing the experiment.

Some important SIMAN IV features include:

1. Special-purpose constructs to simplify and enhance the modelling of manufacturing systems.

2. Constructs which make it easy to model material handling devices such as AGVs and conveyors.

3. Blocks to allow input-output in the model without user-code. These support formatted, unformatted, sequential, direct-access, and spreadsheet files, providing an interface to many products and databases.

4. An interactive debugger which allows you to monitor and control the execution of the simulation. This provides a powerful tool for tracing and controlling model operation and isolating logical errors.

5. The SIMAN IV Environment which is a menu-driven tool that integrates the function of building and running the model and animation, and analysing both input and output data.

6. Transparent use of the CINEMA IV system to generate real-time, high-resolution colour animations of the system dynamics. This provides a powerful tool for both understanding and explaining the dynamics of a system.

7. Complete compatibility across mainframe, workstation, and microcomputers. Models can be moved between computer systems without modification.

2. SIMAN IV environment

The following sections will review the graphical interfaces to SIMAN IV.

2.1 Model preparation

BLOCKS and ELEMENTS are menu-driven, "fill-in-the-form" editors used to build syntactically correct SIMAN IV model and experiment files.

A block diagram model can be defined in either of two equivalent forms referred to as the diagram model and the statement model. The diagram model is a graphic representation of the system using the basic block symbols. These diagrams are linear flowcharts that depict the movement of entities through the system. The statement model is a more "programming-like" representation of the model. Either form can be generated from the other.

BLOCKS allows you to build SIMAN IV models in a graphical, block diagram format. Block diagrams are displayed in model windows on the graphics screen. The experiment frame can be created concurrently using ELEMENTS. Users who prefer to use their own text editor for model and experiment preparation may do so from within the environment while retaining its other benefits.

2.2 Input data analysis

The SIMAN IV Input Processor is a graphical, menu-driven program that assists the analyst in determining the best probability distribution for given sets of data, providing the appropriate expression to use in the SIMAN IV model. It fits a specific distribution to the data, allowing the analyst to compare one distribution with another. The analyst can also display the effects of changing parameters within a given distribution.

2.3 Output data analysis

The SIMAN IV Output Processor is a graphical, menu-driven program to help analyse data generated from simulations.

SIMAN IV generates output files that can be used to generate statistical and graphical measures of performance such as confidence intervals, correlograms, histograms, etc. One output file can be analysed in many different ways without re-executing the simulation program. An analysis can be based on multiple runs of a model or used to compare the response of two or more systems.

2.4 Animation

CINEMA IV is a menu-driven package used to create graphical representations of SIMAN IV simulation models. When the SIMAN IV model and the CINEMA IV layout are brought together, the user is able to view a real-time animation of the simulation model with jobs or customers moving through the system. Cinema builds an animation without any programming effort. An animation can present information in an interesting and understandable format, point out system flaws and problems, describe the system to unfamiliar audiences, and aid in validating and verifying the simulation model.

3. General-purpose modelling features

Models are constructed as block diagrams which depict the flow of entities through the system. The block diagram symbol shapes indicate their function. The sequencing of blocks is depicted by arrows which control the flow of entities from block to block through the entire diagram.

The entities are used to represent workpieces, information, people, etc., that flow through the real system. Each entity may be individualized by assigning attributes to describe or characterize it. For example, an entity representing a workpiece might have attributes named DueDate and ProcessingTime corresponding to due date and processing time for the workpiece. As the entities flow from block to block, they may be delayed, disposed, combined with other entities, etc., as determined by the function of each block.

Each block has operands that control its functions. For example, the CREATE block has operands which prescribe the time between batch arrivals, the first arrival time, the number of entities per batch, and the maximum number of batches to create.

Blocks may optionally be assigned a block label and one or more block modifiers. A block label is used for branching or referencing from other blocks. Block modifiers can modify or extend the standard function to be performed by the block.

To illustrate the general-purpose modelling approach of SIMAN IV, consider the simple manufacturing system in which workpieces arrive, are processed in order on a single machine, and then depart the system (Figure 1).

The block diagram model for this example is shown in Figure 2. The workpieces enter the system at the CREATE block. The operands for this block specify that the workpieces enter in batches of 10 and that the interarrival time between batches is exponentially distributed with a mean of 20. The workpieces continue to the QUEUE block named Buffer where they wait in turn to seize a unit of the resource Machine. Once a workpiece seizes the Machine, it enters the DELAY block where it is delayed by the processing time which is specified as a sample from a uniform distribution between 1 and 2. Following this, the workpiece releases the resource Machine, which allows it to be re-allocated to workpieces waiting in the QUEUE block. The symbol attached to the bottom of the RELEASE block is called the DISPOSE modifier and models the departure of the workpiece from the system.

An example experiment frame for this model is shown in Figure 3. It specifies the conditions associated with the model and includes the definition and capacity of the resources employed, a specification of the number and length of each simulation replication, etc.

4. Manufacturing features

In this section, we will describe the features included in SIMAN IV for modelling the characteristics of manufacturing systems.

4.1 Modelling workstations

Large manufacturing systems typically consist of a number of different workcentres or cells. A natural way to model such systems is to decompose the large system into its workcentres, modelling each workcentre separately, and then combine the workcentre models into an overall system model. This can be done within SIMAN IV by employing the STATION block which defines the beginning of a station submodel. An entity is entered into the station submodel by sending the entity to the STATION block using a transfer block. A transfer block, such as the CONVEY or TRANSPORT block, is used to represent entity movements between station submodels.

Each station submodel is referenced by a station name (e.g. Inspection) and a station number which corresponds to a physical location within the system. The station name or number, which can be used interchangeably, is an operand of both the STATION block and a transfer block.

When an entity enters a STATION block, the entity's station attribute, M, is set to the station number. The entity carries this special attribute with it as it proceeds through the sequence of blocks which comprises the station submodel. The entity remains within the station submodel until it is disposed, or until it is sent to a new station submodel via a transfer block. The block sequence within a station submodel defines the processes through which the entities flow.

To illustrate the concept of a workcentre submodel, consider the block diagram submodel shown in Figure 4. This block diagram contains the frequently occurring sequence QUEUE-SEIZE-DELAY-RELEASE which can be used to model both a single-server and a multi-server queueing system, depending on the capacity for the resource that is specified in the experimental frame.

The workpieces arriving to this submodel enter the STATION block name 'athes, proceed through the QUEUE-SEIZE-DELAY-RELEASE blocks and then enter the ROUTE block. The ROUTE block is a transfer block which routes the workpieces to their next workcentre.

The block sequence in this example is particularly simple and employs only a small subset of the features of SIMAN IV. Once the analyst becomes familiar with the wide range of block functions included in SIMAN IV, complex workcentres can be modelled with similar ease.

4.2 Macro submodels

One particularly useful feature for modelling workcentres in SIMAN IV is the macro submodel. This powerful feature permits the development of a single macro submodel to represent two or more similar yet distinct workcentres. For example, a typical jobshop consists of several different workcentres (lathes, drills, etc.) that are functionally equivalent and differ only in their number and type of machines, buffer sizes, etc. We can model this jobshop by constructing a single macro submodel which represents the process encountered by a job at a general jobshop workcentre. This single macro submodel can then be used to model a jobshop of arbitrary size.

The beginning of a macro submodel is defined by a STATION block. The range of stations represented by the macro submodel is specified as the operand of the block. An entity is entered into the macro submodel by sending it to the STATION block using a transfer block. All entities sent to a station in the specified range of the STATION block are processed as arrivals to the block. Upon entering the STATION block, the station attribute M of the entity is set by SIMAN IV to the station to which the entity was sent.

When a macro submodel is employed, the station attribute is typically incorporated in the operands of one or more of the blocks that follow the STATION block. In this way, the operation of the macro submodel can depend upon the station of the entity. For example, the station attribute could be used to specify a queue number or the entity could be branched within the submodel based on its current station.

4.3 Process plans

As illustrated in the previous example, a workpiece is sent to its next workcentre using a transfer block. However, the transfer block must have some way to determine which workcentre is next in sequence for a particular workpiece. In addition, it may be necessary to update one or more attributes of the workpiece to correspond to the processing parameters at that workcentre.

The workcentre visitation sequence and corresponding attribute update values are specified in SIMAN IV using the SEQUENCES element.

The SEQUENCE element permits defining steps in a process plan as well as defining the data associated to the plan such as setup times, special tool requirements, etc. In addition, a step of the process plan could be repeated or skipped, or an alternate process plan could be followed.

4.4 Resource schedules

The workcentres within a manufacturing system often operate according to different work schedules as a result of their differing loads. Within SIMAN IV, this can easily be modelled through the use of the SCHEDULES element. This element is used to define a work schedule by specifying a resource capacity over time. A resource capacity within the model can then be directed to follow a given work schedule. For example, the resources in workcentre 1 might be directed to follow schedule number 1 and the resources in workcentre number 2 might be directed to follow schedule number 2.

The SCHEDULES element shown in Figure 5 defines two different work schedules.

In schedule number 1, the capacity is 1 for 8 time units, then 0 for 16 time units, and then this cycle repeats. In schedule number 2, the capacity is 1 for a duration that is sampled from an exponential distribution, and then 0 for a duration that is sampled from a uniform distribution, and then this cycle repeats. Note that schedules can be used to represent breakdowns and repair activity for a resource such as a machine.

4.5 Modelling material handling systems

Within a manufacturing system, the movement of entities between workcentres is accomplished by the material handling system. This is an extremely critical function in most manufacturing systems that can easily account for a significant per cent of the production activity.

The categorization of material handling equipment into the two basic movement functions of transport and convey provides the basis for modelling these devices in SIMAN IV. The transport function corresponds to the intermittent movement of items, one load at a time, along a fixed or varied path. The term load as applied here could denote a box, a roll of material, or a pallet containing a number of items grouped together. The convey function corresponds to the continuous movement of items along a fixed path. Special blocks and elements are included in SIMAN IV that allow both of these movement functions to be modelled in a straightforward manner.

The blocks that are used to model material handling systems employ the concept of a station submodel as discussed earlier. All movements are made relative to station names assigned to each station submodel. The travel time for entities between stations is based on the speed of the material handling device and the spatial relationship of the origin and destination stations relative to the device. Both of these are specified by the modeller in the experimental frame.

4.5.1 Transporters

The generic term transporter is used in SIMAN IV to denote a general class of movable devices that may be allocated to entities. Examples of devices which might be modelled as transporters are carts, cranes, and mechanical manipulators.

There are two different types of transporters in SIMAN IV. The first, standard transporters, are unconstrained in their movement between stations. These are useful in modelling situations where transporter traffic congestion is minimal. The second, called guided transporters, are used to model situations where the transporters are constrained to moving over a

defined network consisting of links and intersections. Guided transporters are particularly useful for modelling automatic guided vehicles (AGVs) and automated storage and retrieval systems (AS/RS).

The characteristics of each transporter type are specified in the experimental frame and include the name, capacity, system map, and the initial station position and operational status of each of the transporter units for that type. The capacity is the number of independent movable units of that transporter type. The system map is a cross-reference to a definition of the feasible travel paths and associated travel distances between pairs of stations which each transporter unit of that type may visit. In the case of guided transporters, the system map includes a description of the network of links and intersections.

Transporter units are allocated to entities at a REQUEST block, after which the entity can be transported from one station to the next using the TRANSPORT block. The duration of the transport is automatically computed by SIMAN IV based on the distance to the station and the speed of the transporter unit. For guided transporters, the travel time may be further influenced by other traffic in the system. At the end of the transport duration, the entity enters the STATION block of the destination station submodel.

4.5.2 Conveyors

The generic term conveyor is used in SIMAN IV to denote a class of devices which consist of positioned cells linked together that move in unison. Each cell represents a location on the device and can be either empty or occupied. Entities that access the conveyor must wait at the entering station until the specified number of consecutive empty cells are located at the station. The entity then enters the conveyor and the status of the cells are changed from empty to occupied. The entity remains in the cells until the conveyor is exited at the entity's destination station.

Each conveyor in SIMAN IV can be defined to be either accumulating or non-accumulating. In an accumulating conveyor, an entity that is stopped on the conveyor forms a blockage point for other entities that continue to move on the conveyor. Entities arriving to the blockage point accumulate behind the blockage. In a non-accumulating conveyor, an entity that stops on the conveyor forces the entire conveyor to stop. As a result, there is no accumulation of entities.

Each conveyor device moves along a fixed path defined by one or more segments. A segment is a section of a conveyor path that connects two station submodels. Segments can be connected to form either open- or closed-loop conveyor paths. A closed-loop path is one in which an item on the conveyor can return to a station by continuing on the device. An open-loop path is one that is not closed. The segments defining a conveyor path are specified in the experimental frame.

4.6 Modelling shop floor control

A rapidly growing application of simulation is finite capacity scheduling. Simulation is a powerful tool for scheduling. Although there are simulation-based packages targeted specifically at scheduling, a general-purpose simulation language has several advantages:

(a) Models developed and verified during system design can be reused during later analysis and control;

(b) The same simulation language can be used through all modelling phases;

(c) It is easy to accurately model complex systems;

(d) You have the benefit of powerful animations to aid in user training and acceptance.

SIMAN IV contains many constructs that aid in model initialization, database access, report generation, defining Process Plans, JIT, and advanced manufacturing constructs.

Systems Modelling is constantly improving the use of SIMAN IV for Shop Floor Control and is currently developing a product specifically oriented toward scheduling applications.

5. Summary

Since its introduction, SIMAN has been used in a wide range of applications by numerous companies throughout the world. Although many of the applications have been simulations of manufacturing systems, SIMAN has also been applied to general system simulation including health care, computer, and retail systems. SIMAN IV is continually enhanced to retain its position as the state-of-the-art in simulation.

In this paper, we have given only a brief overview of the modelling features of SIMAN IV with an attempt to highlight those features which are particularly relevant to manufacturing systems. Only a small subset of the block functions were discussed, and no attempt was made to describe the enhanced general-purpose features included in the language.

Acknowledgements

This tutorial was adapted from the "Introduction to SIMAN" tutorial prepared by D. T. Sturrock and C. D. Pegden for the Proceedings of the 1990 Winter Simulation Conference.

References

Pegden, C. D., Shannon, R. E. and Sadowski, R. P. (1990). Introduction to Simulation using SIMAN. McGraw-Hill, Inc.

Systems Modeling Corporation (1990). SIMAN IV reference guide, Sewickley, Pennsylvania.

Systems Modeling Corporation (1990). CINEMA IV reference guide, Sewickley, Pennsylvania.

Systems Modeling Corporation (1990). The SIMAN IV environment reference guide, Sewickley, Pennsylvania.

Zeigler, B. P. (1976). Theory of modeling and simulation. John Wiley.

Authors' biographies

Cynthia J. Kasales is currently an engineer performing consulting functions for the Simulation Services Division of Systems Modeling Corporation. She has developed models in a wide range of application areas such as transportation and wafer fabrication. She received a BS in Industrial Engineering from the Pennsylvania State University. Prior to joining the Simulation Services Division, Ms. Kasales was on the Technical Support Staff where her duties included providing end user support, software testing and porting.

David T. Sturrock is currently Development Project Manager for SIMAN IV with Systems Modeling Corporation. He received a BS in Industrial Engineering from the Pennsylvania State University, with concentrations in manufacturing and automation. Prior to joining Systems Modeling, he worked in several manufacturing facilities including more than ten years at Inland Steel Company. He has applied simulation techniques to problem solving in the areas of transportation systems, scheduling, plant layout, capacity analysis, and process design. He is a member of IIE, SME, CASA, and SCS.

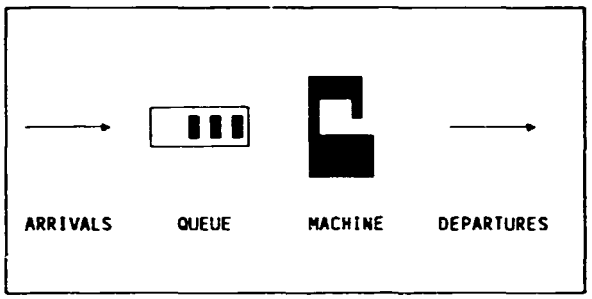


Figure 1: Schematic of a Simple System

```
Begin;
Project,Simple System,Analyst;
Queues: Buffer;
Resources: Machine, 1;
Replicate, 1, 0, 480;
End;
```

Figure 3 : Experiment for Simple System

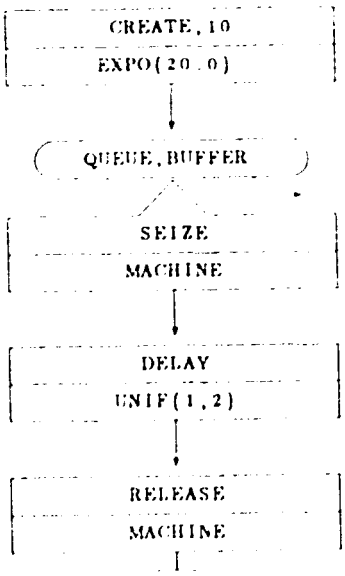


Figure 2: Block Diagram for Simple System

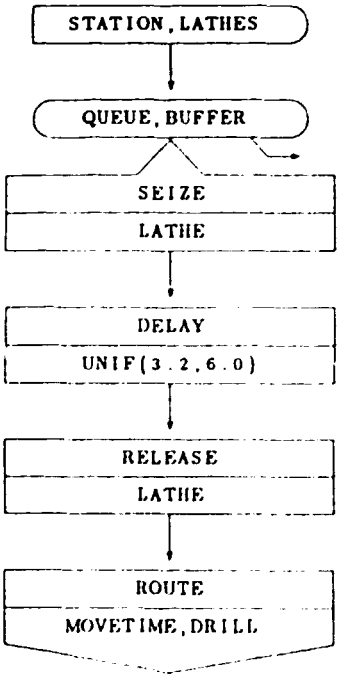


Figure 4: Block Diagram of a Workcenter

```
SCHEDULES:
 1, 1*8, 0*16:
 2, 1*EXPO(5.5), 0*UNIF(.5,1);
```

Figure 5: SCHEDULES Element

7. COMPUTER ANIMATION WITH CINEMA

David R. Kalasky
Deborah A. Davis

Systems Modeling Corporation
The Park Building
504 Beaver Street
Sewickley, Pennsylvania 15143

Abstract

CINEMA is a general-purpose animation system designed to animate models developed using the SIMAN simulation language. The CINEMA package consists of a module used to draw the graphical images used in the animation and a module to execute a SIMAN model and associated animation.

Animation typically centres around the presentation of final modelling results. However, the simplicity of developing CINEMA animations allows users to exploit animation in more phases of a simulation project. By using CINEMA, animation benefits can be found throughout the entire simulation project from initial model building through the presentation of the final simulation results.

CINEMA IV complements the advanced capabilities found in SIMAN. While any process can be modelled using SIMAN/CINEMA, the software has exceptional ease and accuracy in modelling material handling functions such as accumulating conveyors and AGV systems. CINEMA provides real-time graphical outputs for SIMAN statistical routines such as dynamic plots and histograms.

CINEMA is available on a wide variety of platforms and operating systems. These include microcomputers under DOS and OS/2, as well as Apollo, IBM, Sun, and VAX workstations under the AIX Windows, UNIX, and VMS operating systems.

1. Introduction

In recent years animation has become a requirement of the simulation process. One of the reasons for this requirement is that numeric summary statistics do not necessarily convey information about the dynamic interactions of components of a system. Although summary statistics are a crucial part of evaluating the performance of a simulated system, it is only through animation that the analyst can easily identify the system status under which, for example, bottlenecks occur. Using the general purpose simulation language SIMAN (Pegden, 1990), in conjunction with the CINEMA animation system, offers a means of analysing both the system performance and the dynamics during the simulation of any discrete-event system.

Benefits of animation are not restricted to bottleneck analysis only. Johnson and Poorte (1988) identified four simulation modelling phases where the modeller may benefit from using animation. They are:

1. Model building and verification;
2. Model validation;
3. Bottleneck analysis;
4. Communication and presentation.

1.1 Model building and verification

Animation provides a tool to visually inspect that a model acts and reacts as intended. By viewing the animation, logical errors are easily identified. This added view of the model in action reduces the likelihood of inaccurate modelling. Therefore, invalid conclusions, incorrect decisions, and large failure costs may be avoided.

1.2 Model validation

Validation requires thorough knowledge and understanding of the intricacies of the system under consideration. The analyst may have the required awareness but by using simulation only, one can never be totally sure that the model is a good representation of the real system. Modelled projects require input and review by several different people. The reviewers typically have varying degrees of modelling and process expertise. Animations provide a common database of communication concerning the systems being evaluated.

Animation provides the analyst with a powerful tool to ensure that the model is indeed a sufficiently adequate representation of the real system. For example, the impact that a simplification may have on the model is readily seen in a running animation. Assumptions that have a profound effect on the performance of a model may go unnoticed without the use of animation, which, again, could lead to inaccurate results and incorrect conclusions. As a model validation tool, animation allows the modeller to make reasonable simplifications and other model assumptions.

1.3 Bottleneck analysis

As an analysis tool animation can play a major role in bottleneck analysis. Using animation, the modeller has the ability to observe interactions of several simultaneous and interrelated events, thus providing information unavailable in aggregate statistical performance measures. Bottleneck situations are usually simple to locate with animation. Also, the important preconditions under which these and other unfavourable situations occur can be identified. By observing an animation, these factors become apparent and they may also help visualize candidate improvements.

1.4 Communication and presentation

The ultimate objective of any simulation project is to provide credible information to the decision maker. Credibility of a project is strongly associated with the ability of the modeller to effectively communicate throughout a simulation project. Animation gives the analyst a very powerful vehicle to provide the nonsimulationist with insight into a complex

model. Likewise, a "process expert" can visualize and appreciate the simulation methodology.

It should be stressed that animation cannot replace standard statistical analysis techniques. However, the above discussion illustrates that with animation the modeller has access to a powerful analysis tool. As such, animation has the ability to effectively change and enhance virtually all phases of the modelling process.

2. Design philosophy of CINEMA

Designing and building realistic simulation models is a complicated and often time-consuming process. CINEMA was designed for the user to exploit animation throughout a simulation project from initial model building through the presentation of the final results. Certain features of the architecture of the CINEMA animation package allows for this versatility.

Three main design objectives for CINEMA were:

1. Simplicity;
2. Flexibility;
3. Effectiveness.

CINEMA's simplicity of use permits easy mastery of the task of building an animation; its flexibility allows graphical animation of any system at any level of detail; and its effectiveness greatly assists in communication throughout a simulation study.

2.1 Simplicity

A common objective for software architects is that their products be easy to use. One key element in the design of CINEMA, is its ease of use; no special training is required to readily construct impressive animations. A mouse-controlled user-interface where users select items from pop-down menus is the way in which animation layouts are developed. With CINEMA, no programming is required to build an animation. The user can fully concentrate on developing useful, realistic models and the data to support the analysis.

2.2 Flexibility

The CINEMA animation package is intimately tied to the SIMAN simulation language. SIMAN is a general purpose language and has been used in a wide variety of environments, such as manufacturing, communication, health care, and defence. CINEMA is used in the same environments, therefore the same flexibility is incorporated in the animation.

Johnson and Poorte (1988) have proposed a hierarchical approach to computer animation in simulation modelling. Under their approach, animation is used during all phases of a modelling project. Animations evolve from simple modelling tools, to interactive analysis tools, until the final presentation of the results as a powerful communication vehicle. The associated animations are of increasing visual accuracy and detail. The authors support CINEMA as unique in providing the flexibility necessary for a multi-level approach to computer animation.

2.3 Effectiveness

Effectiveness of an animation system is measured by how valuable the graphical images are during the different phases of a simulation project. CINEMA's design assures effective use of animation. It can play the role of a verification and validation tool in the early phases of a project. In the next phase CINEMA becomes a powerful analysis tool in which bottlenecks can be detected and many "what-if" questions can be explored interactively. In the final phase of a project, impressive animations can be constructed with CINEMA and it becomes a communication tool highlighting key features for targeted audiences.

3. Building a CINEMA animation

The user interacts with CINEMA by using a mouse-controlled graphics cursor to pick items from pop-down menus as shown in Figure 1. Context sensitive help is available on-line simply by pointing to an item and clicking the right mouse button. The left mouse button activates the currently highlighted function.

The CINEMA layout is a graphical representation of the system being simulated. The layout consists of graphical objects grouped into two categories: static and dynamic. The static component of the layout does not change during the execution of the simulation, while the dynamic component consists of graphical images which do change colour, location, and shape.

3.1 The static component

The static component of a layout represents the physical environment in which the simulated system exists. It includes anything that helps to visually recognize the scene, but does not change during model execution. The static component (called background in CINEMA) can be created by using the drawing facilities provided in the CINEMA program. Under the draw menu drawing functions are available for line, polyline, box, circle, bar (a filled box), arc, and freehand sketch. Drawing options can be modified by selecting a different colour, line style, and/or line thickness for each drawing function. Text can be added to the background at any location and in any orientation. Text options include any combination of six different sizes, with six different fonts, and 16 active colours. Explode allows the user to zoom in for fine detailing of small portions of the drawing, while Cut Area provides a means of copying or moving rectangular regions of the background or saving regions to files for use in other CINEMA animations.

An alternate method for generating a background is to start with a DXF-formatted file generated by a computer-aided design (CAD) packages such as AutoCAD, PC-CAD, CADVANCE, etc. Many of these packages have sophisticated drawing functions, especially for three-dimensional images, allowing the user to develop geometrically accurate and more complex backgrounds. DXF-formatted files generated on any hardware platform can be input into CINEMA. An example of a layout generated via this CINEMA utility is illustrated in Figure 3.

By using CAD drawings and CINEMA the user obtains the best of both worlds. For example, AutoCAD has powerful 3-D drawing commands, whereas CINEMA contains functions that cannot be performed by CAD packages, such as area fills and pixel-based editing. Impressive displays have been generated by using AutoCAD to develop the "technical part" of the layout followed by CINEMA to fine tune the "aesthetic part" of the layout.

3.2 The dynamic component

The dynamic component of the layout consists of graphical images which change shape, colour, size, or location in response to a status change in the SIMAN model. The dynamic objects in a CINEMA layout are directly tied to SIMAN modelling constructs. As the state of the constructs change during execution, the associated dynamic objects will change to reflect this change of the SIMAN construct. A large number of dynamic objects are available; each one is discussed in some detail below.

Entities - Entities represent the items that are being processed or flow through the system. Entity symbols are created by drawing the icon on a blown-up grid ("fat bits"), using the mouse-controlled cursor. As the symbol is created on this grid, it is simultaneously displayed in actual size in the upper left corner of the screen, as shown in Figure 3.

In CINEMA, a specific symbol from an entity library is associated with a unique attribute number. When the animation attribute changes values in the SIMAN model, the corresponding entity symbol changes in the animation. Consider, for example, a simulation of an automotive assembly plant. The entity arriving to an assembly station might have a value that corresponds to a symbol of a car body without doors (Figure 3). After leaving the workstation, the symbol could be changed to a car body with doors (Figure 4) simply by reassigning the symbol number in the SIMAN model.

Resources - Resources are used in SIMAN to model limited items in a system, such as machines and workers. Like entity symbols, resource symbols are created by drawing icons on the enlarged grid with the mouse. Resources dynamically change in a system between idle, busy, inactive, or pre-empted. Resource status changes within a SIMAN model are displayed in CINEMA animation by using resource symbols that represent the resource in each of the possible states.

Queues - A queue can be added to the layout at any location, and in any length and orientation representing a SIMAN QUEUE block. These entities might represent workpieces awaiting the availability of a machine, a set-up operator, cars at a door assembly station, etc. When an entity enters a queue in the SIMAN model, the entity symbol is displayed along the corresponding queue symbol at the proper location relative to the other members of the queue. When an entity exits the queue in the model, its associated symbol is removed and all following symbols are moved forward one position. Sorting and prioritization can occur within queues and viewed via CINEMA.

Transfers - CINEMA's transfer menu is used to define the paths of travel for ROUTES, CONVEYORS, and TRANSPORTERS. STATION symbols are used to designate start and end points for the movement of entities. One method for modelling the movement of entities between STATIONS is to use the ROUTE block. When an entity reaches a ROUTE block in the SIMAN model, its entity symbol is continually redisplayed at new points along a predefined CINEMA route path to produce the effect of movement.

Movement that involves SIMAN material handling constructs is designated in much the same way as the routes. If entities are transferred by a CONVEYOR, the user will digitize paths called segments, and if the entities are moved by TRANSPORTERS, the paths will be called distances. All route, segment and distance paths are transparent during model execution, yet the entity symbols will trace these paths as they move.

Variables - While a simulation is executing, SIMAN automatically maintains the value of status variables which define the system state. Examples of these status variables are: simulation time, queue values, throughput, resource utilization, etc. Any SIMAN status variable can be incorporated into an animation layout using one of five dynamic features in CINEMA.

1. Digital displays can be added to a layout using a feature called dynamic variables. Users can tailor the format for display, range of values, size, and colour using pull-down menus.

2. A second way to display status variables is with an analog representation of a variable's value. Three different level shapes are included in CINEMA: a box, a circle, and a dial. During execution of an animation, box or circle levels fill and empty in response to changes in the value of the associated status variable. The dial is a circular level with a sweep hand that rotates either clockwise or counter-clockwise. Dials are typically used to represent simulated time with a clock.

3. A feature called global symbols represents a third way to display the value of a status variable. Like entity and resource symbols, global symbols are drawn on the enlarged grid. For example, a symbol saying "STARVED" could be displayed when the number in a queue for a machine is zero, and a second symbol saying "BLOCKED" could be displayed when the queue is full to capacity.

4. Dynamic colours represents a fourth way to display the value of a status variable. While global symbols indicate specific conditions, dynamic colours can be assigned to indicate gradual state changes.

5. Plots and Histograms can also be displayed on the animation to show status variables changing over time.

4. Running an animation

Discussion so far has focused on the capabilities of the layout generation module of CINEMA. The execution of the SIMAN model in the CINEMA package has the ability to update the dynamic component of the layout interactively during execution as well as control the speed of execution. Since CINEMA is a real-time animation system, as opposed to a post-processed system, the menu includes facilities to temporarily stop a run and use the SIMAN interactive features. The interactive mode allows a user to change variables, look at the entities in a particular queue, look at status variables not displayed on the animation, and monitor practically any variable in the system. A user can turn on the SIMAN trace function so that the animation can be viewed simultaneously with the model execution statements. This feature is very useful for model verification and validation. A stepping facility halts the model execution after each event is processed. Execution is resumed when the user presses the space bar and continues until the time the next event is processed.

The user may associate any number of layouts with a single SIMAN model. These layouts are identified in files and may be recalled from the menus or directly from the keyboard. Figure 5 shows three layouts for a single SIMAN model. The first shows the entire facility while the other two layouts show detail of two different work centres. Another CINEMA feature is to save a snapshot: This saves a picture of the layout at a specific instant in simulated time as well as saving the value of all the system variables. The snapshot may be recalled at a later time so that the simulation can progress from the time that the snapshot was saved. This feature is particularly useful in demonstrating and presenting critical situations and comparing variations of the system at the same moment in time.

5. Multi-platform support

Since the initial release of CINEMA in 1986, many new platforms have been added. The primary platform has been the IBM PC-AT (or compatible) and the DOS operating system. There are currently three graphic cards available for the PC-AT/DOS configuration, namely the EGA card (or VGA card in EGA mode), a high-resolution card (HGA) purchased through Systems Modeling, or the IBM 8514/A card. The EGA has a pixel resolution of 640x350 while the high-resolution cards (HGA and 8514) have a pixel resolution of 1024x768. OS/2 allows simulation of systems virtually unlimited in size.

CINEMA is available on the Apollo DN3000 series workstation, Digital Equipment Corporation (DEC) VAXstation series of workstations (including MicroVAX workstations with the GPX upgrade), IBM RS-6000 workstations, and Sun Microsystems Sparc series workstations.

6. Summary

CINEMA is a general purpose animation system which allows for the easy animation of any SIMAN simulation model. Because CINEMA utilizes user constructed icons, any type of system ranging from manufacturing, to distribution, to health care, transportation, and communications may be animated.

Three main design objectives of CINEMA are: (1) simplicity, (2) flexibility, and (3) effectiveness. The mouse-oriented, menu-driven user interface allows for the rapid development of animations without the need for programming. CINEMA animations aid the analyst in the process of building and verifying SIMAN models. Other benefits of CINEMA can be found during model validation and analysis phases, as well as in the presentation of the final results.

Acknowledgements

This tutorial was adapted from the "Computer Animation with Cinema" tutorial prepared by J. P. Poorte and D. A. Davis for the Proceedings of the 1990 Winter Simulation Conference.

References

Johnson, M. E. and J. P. Poorte (1988), "A hierarchical approach to computer animation in simulation modelling. Simulation 50, 1, 30-36.

Kasales, C. J. and D. T. Sturrock (1991), "Introduction to SIMAN", in Proceedings of the 1991 Winter Simulation Conference, B. L. Nelson, W. D. Kelton, and G. M. Clark, Eds. IEEE, Piscataway, New Jersey.

Pegden, C. D., R. E. Shannon, and R. P. Sadowski (1990), Introduction to simulation using SIMAN, McGraw-Hill, New York, New York.

Poorte, J. P. and D. A. Davis (1990), "Computer Animation with Cinema", in Proceedings of the 1990 Winter Simulation Conference, O. Balci, R. P. Sadowski, and R. E. Nance, Eds. IEEE, Piscataway, New Jersey, 123-127.

Author biographies

David R. Kalasky is the Product Services Manager for Systems Modeling Corporation. Prior to joining Systems Modeling, Mr. Kalasky was Manager of Simulation for Westinghouse at their Corporate Productivity and Quality Center. Mr. Kalasky holds a B.S. degree in Industrial Engineering from the University of Kentucky and an M.S. degree in Operations Research from the Pennsylvania State University. His experience includes work in engineering and the management of production and quality as well as simulation consulting. He is a registered Professional Engineer, an active member of IIE and SCS, CPIM and instructor within APICS, invited author and speaker for SME, and was Business Chairman for the 1990 Winter Simulation Conference.

Deborah A. Davis is the Vice-President of Software Development with Systems Modeling Corporation (SM). Since joining SM in 1984, she has worked on development of SIMAN, CINEMA, and other SMC products. Miss Davis received B.S. and M.S. degrees in Industrial Engineering and Operations Research from the Pennsylvania State University. Her current interests include simulation language and user interface design, new applications of simulation technology, and object oriented methodologies. She is currently the Business Chair for the 1991 WSC, was the Exhibits Chair for the 1990 WSC and will be the General Chair of the 1994 WSC. She is a member of SCS, IIE, Tau Beta Pi, and Alpha Pi Mu.

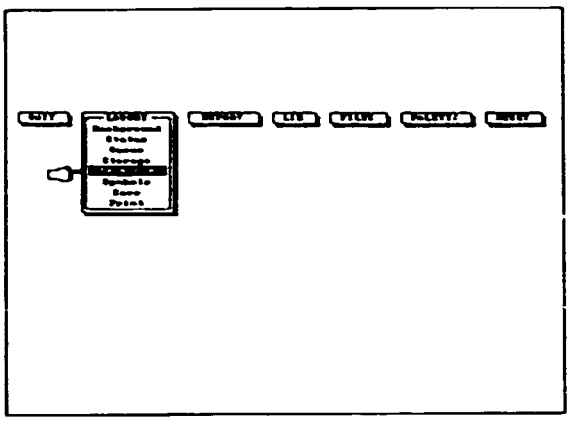


Figure 1: The CINEMA Interface

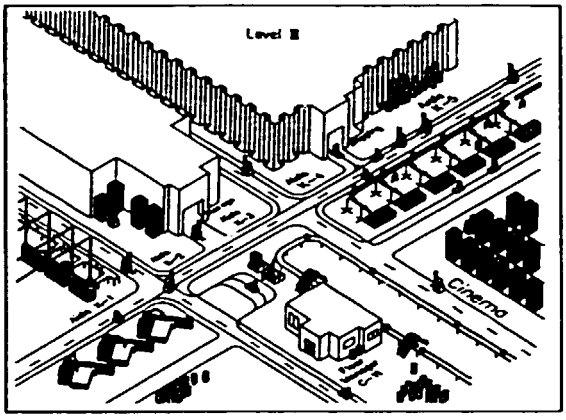


Figure 2: CINEMA Layout Imported From a CAD Drawing.

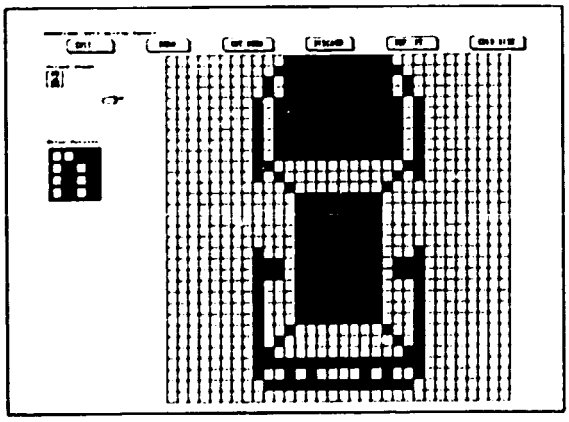


Figure 3: Creating an Entity Symbol

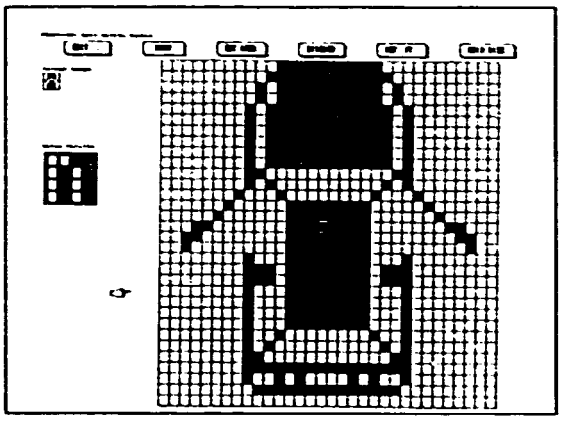


Figure 4: Subsequent Entity Symbol

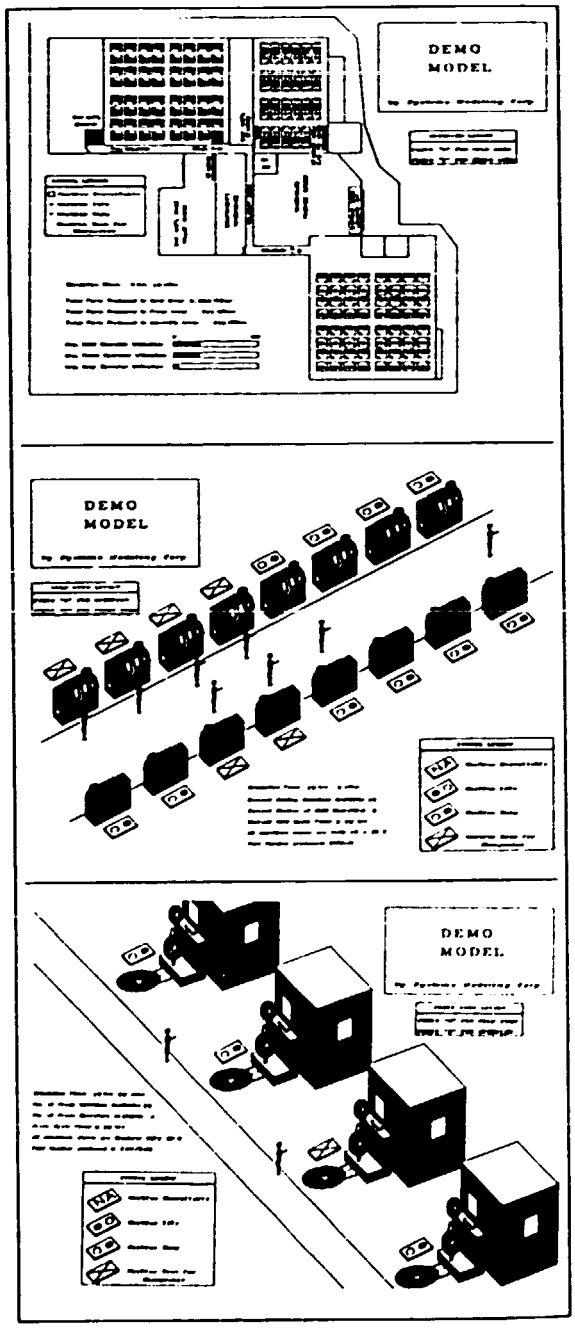


Figure 5: Multi-Layout Animated Model

3. SLAM II® TUTORIAL

Jean J. O'Reilly

Pritsker Corporation
1305 Cumberland Avenue
West Lafayette, Indiana 47906

Abstract

SLAM II was the first simulation language which allowed a modeller to formulate a system description using process, event, or continuous world views or any combination of the three. Since its initial release in 1981, SLAM II has undergone continual development and application. This paper will provide an introduction to the modelling language and describe the most recent developments in SLAM II.

1. Introduction

SLAM II, the Simulation Language for Alternative Modelling, was the first simulation language which allowed a modeller to formulate a system description using any of three approaches (world views) or any combination of the three. This integrated framework allows the SLAM II user to take advantage of the simplicity of the process-oriented (network) approach and to extend a model with discrete event constructs should the network approach become too restrictive. Continuous variables may be used in conjunction with a network or discrete event model whenever this is the most convenient way to represent system elements. The ability to construct combined network-discrete event-continuous models with interactions between each orientation makes SLAM II an extremely flexible tool for simulation.

Since its introduction, SLAM II has continued to evolve as a result of extensive application. Experience with thousands of models has demonstrated the flexibility of the language, but has also pointed out the ways in which SLAM II could be extended for greater ease of use.

In addition to enhancements to the modelling language itself, the following software has been developed for use with SLAM II:

- TESS (The Extended Simulation System) provides database management for simulation output data and facilities for graphically building models, and analysing, graphing, and animating model results;
- A Material Handling Extension (MHEX) provides detailed modelling of regular and special resources (cranes, storage areas, automatic guided vehicles, and guidepaths);
- The SLAM II Interactive Execution Environment (IEE) allows the interruption of a simulation in order to examine system status, reassign variable values, step through events, or call SLAM II support routines for debugging or gaming;
- SLAMSYSTEM includes SLAM II in an integrated simulation system for advanced personal computers.

This tutorial will introduce some of the modelling techniques used in SLAM II, illustrated with simple examples.

2. Network modelling

A simulation model normally begins with a network, or flow diagram, which graphically portrays the flow of entities (people, parts, or information, for example) through the system. A SLAM II network is made up of "nodes" at which processing is performed. SLAM II nodes, shown in figure 1, provide for such functions as entering or exiting the system, seizing or freeing a resource, changing variable values, collecting statistics, and starting or stopping entity flow based on system conditions. Nodes are connected by branches, called "activities", which define the routing of the entities through the system. Routing may be deterministic, probabilistic, or based on system variables. Time delays on activities may represent processing times, travel times, or waiting times. Entities which proceed from node to node over activities may have unique characteristics, all "attributes", which control their processing. Entities may reside in "files", or ordered lists of entities which are waiting for some change in system status. The graphical framework for representing a network model simplifies model development and communication.

The process of building a SLAM II network model consists of choosing the symbols which can represent system processes, combining them in a diagram which represents the entity flow, and parameterizing the symbols with model-specific data. A single-server queueing model (representing, for example, a workstation) is shown in figure 2. The network begins with a CREATE node which generates the first job arrival at simulated time 0.0 and continues to generate arrivals at a rate drawn from an exponential distribution. A QUEUE node is used to delay arrivals until the station is available. The station, whose processing time is sampled from a normal distribution, is represented by the ACTIVITY, or branch, following the QUEUE. Upon completion of the activity, a COLCT node records the interval between departure time and the job's arrival time, which was stored in attribute 1. The graphic modelling approach is both quick to use and an effective way to communicate the structure of a model.

Unless the network was constructed using TESS or SLAMSYSTEM, the diagram is then translated into a set of input statements as shown in figure 3. Each symbol corresponds to an input statement, and each statement may be followed by a comment which describes the processing being performed. The output from this model would automatically report statistics on job waiting time, queue length, station utilization, time in system and throughput.

3. Using discrete event concepts

In the discrete event orientation of SLAM II, the modeller identifies the discrete points in

time at which the state of the system can change and develops the logic associated with each such "event". SLAM II provides support subroutines which perform such common simulation tasks as scheduling events, moving entities into and out of files, collecting statistics, and obtaining random samples. Most models built with SLAM II are not strictly network or discrete event but a combination of the two approaches.

Several interfaces are possible between a SLAM II network and user-written inserts. One is the EVENT node, which is a "do-it-yourself" node. The EVENT node invokes a user-written subroutine in which highly complex logic may be performed. Support subprograms provide information on system status and allow that status to be changed. Other interfaces to user-written logic provide for complicated variable calculations and sophisticated resource allocation logic.

4. Continuous modelling

In a continuous simulation model the state of the system is represented by variables that change continuously over time. The modeller specifies equations which determine the values of state variables and the "step size", or time increment, between the updating of variable values. These equations may be differential equations, in which case the simulator uses a numerical integration algorithm to obtain new variable values from the derivative values.

Continuous variables have proven to be an efficient way to model high-speed, high volume systems such as packaging lines (O'Reilly, 1985). In such a system, a buffer area between two machines may contain several hundred items, too many to be modelled individually. The population of such a buffer is conveniently modelled as a continuous variable which increases at the production rate of the feeding machine and decreases at the production rate of the following machine (figure 4). The equations defining the rates of change for continuous (SS) variables are written in a FORTRAN subroutine. SLAM II updates the variable values at prescribed time intervals and monitors those variables against any threshold values defined. One threshold value, for example, would be the capacity of a buffer. When it is crossed, the feeding machine would need to cease production until the buffer level decreased enough to accept more production.

5. Material handling movements

Among the most complicated elements to incorporate in a simulation model are automated devices which follow fixed paths. These include overhead cranes, stacker cranes, and AGVS (automated guided vehicle systems). Movements of such devices must be modelled in detail if one is to take into account interference among devices which share a common path. When contention occurs, some way must be found to determine which vehicle will be allowed to proceed. A Material Handling Extension (MHEX) to SLAM II, first available in 1986, provides constructs for simulating these complexities (Pritsker, 1986). Its concepts were derived from several simulation models developed at Pritsker Corporation which required detailed material handling logic.

5.1 Modelling cranes

An example involving stacker cranes is shown in figure 5. The schematic depicts a local ASRS system with two stacker cranes serving a lathe and a mill; storage is maintained in nine racks along the crane runway.

The MHEX software takes into account the following complications in this system:

1. Movement times are dependent on crane velocity, distance from destination, and interference with the companion crane;
2. Competing requests for a crane must be prioritized;
3. Storage is limited, and the amount to be allocated depends on the size of an item;
4. If alternative storage locations are possible, selection may be based upon both proximity and material type.

These interactions are illustrated in the network segment shown in figure 6. It begins with a GWAIT (generalized AWAIT) node which requests an available rack and crane. Knowing from the RACK definition (not shown) the capacities and locations of the storage areas and where to find the size of the item to be moved, the software will allocate the closest storage location having sufficient space.

Knowing from the CRANE definitions the velocity, acceleration and deceleration of the equipment, and keeping track of both cranes' positions, the software will release the item only when an available crane (CR1 or CR2) can reach the pick-up point.

After the item is loaded on the crane, taking 0.5 minutes, a GFREE node releases the pick-up point and initiates the crane move. Movement time is calculated internally and is based on equipment speed, distance between the ENTRY and RACK locations, and any interference encountered dynamically. Following transport, 0.5 minutes are required to remove the item from the material handling equipment, and a second GFREE node releases whichever crane was assigned.

5.2 Modelling an AGVS

An Automatic Guided Vehicle System (AGVS) consists of a fleet of vehicles, a guidepath, and a computer control system which determines how a vehicle is selected and routed to a job request. Unlike cranes on runways, AGV's on guidepaths may turn corners and select alternative segments, greatly complicating the logic required to deal with interference and possible alternate routes.

MHEX includes constructs for defining an AGV fleet (number of vehicles, their sizes and speeds) and guidepaths (number of control points, length of each segment, and direction of travel). Once these elements are defined, three node types are used to model the control logic of the system by allocating a vehicle, initiating a move, and releasing a vehicle for reallocation. (Sale, 1987).

6. Interactive execution environment

The SLAM II Interactive Execution Environment provides an interactive user interface to the simulation of a SLAM II model. The modeller may examine, modify, save or load the current system status using the IEE.

The IEE aids a model developer in debugging a model under construction and verifying the completed model. The analyst can use the IEE to develop and analyse alternative control strategies for the system. The what-if questions that arise during model development can be immediately explored using the IEE.

The modeller communicates with the IEE by issuing commands. The commands include:

ADVANCE	HELP
BREAKPOINT	LOAD
CALL	SAVE
CANCEL	SET
CONTINUE	STATUS
DIARY	STEP
EXAMINE	STOP
	TYPE

Using the ADVANCE and STEP command the modeller can control how long the model is simulated. With BREAKPOINTS the modeller can simulate the model until a certain state is reached. For example, one could simulate until the number of orders waiting for processing is greater than six. Using the EXAMINE and SET commands system variables can be viewed and modified.

The IEE is an interactive interface to SLAM II that supports a complete on-line help system. The features of the IEE are fully described in the SLAM II Quick Reference Manual (Pritsker, 1990).

7. Conclusion

SLAM II is a proven, powerful modelling methodology. It has been used for hundreds of simulation projects and as the basis for simulation courses in many colleges and universities. Published applications (see references) describe models dealing with problems in manufacturing, transportation, material handling, staffing, experimental design, communications systems, and many more.

Continuing development of SLAM II and simulation support software has culminated in TESS and SLAMSYSTEM, integrated simulation systems for workstations and personal computers. SLAM II, TESS and SLAMSYSTEM are distributed by Pritsker Corporation, which offers regularly scheduled training classes as well as applications support.

References

Blackwell, R. (1986), "A Discrete Event Scheduler in a Dynamic Production System", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, NJ, 661-664.

Clark, T. D. (1986), "A Systems Analysis and Model of Driver Licensing in the State of Florida", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, NJ, 842-849.

Dessouky, M. M., F. H. Grant and D. Gauthier (1985), "Simulation of an Injector Plunger Production Line", in Proceedings of the 1985 Winter Simulation Conference, D. T. Gantz, G. C. Blais, and S. L. Solomon, Eds., IEEE, Piscataway, NJ, 303-307.

Duket, S. D., and C. R. Standridge (1983), "Applications of Simulation: Combined Models", in Modeling, Issue No. 19.

Erdbruegger, D., D. Starks and C. Farris (1982), "SLAM II Model of the Rose Bowl Staffing Plans", presented at the Winter Simulation Conference, San Diego, California.

Felder, R. M., P. M. Kester and J. M. McConney (1983), "Simulation/Optimization of a Specialities Plant", in Chemical Engineering Progress, July, 25-35.

Garcia, A. B. and W. H. Shaw (1986), "Transient Analysis of a Store- and Foreward Computer-Communications Network", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, New Jersey, 752-759.

Godziela, R. (1986), "Simulation of a Flexible Manufacturing Cell", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, New Jersey, 621-627.

Gross, J. R., S. M. Hare, and S. Roy (1982), "Simulation Modeling as an Aid to Casting Plant Design for an Aluminum Smelter", presented at the IMACS Conference, Montreal, Canada.

Haider, S. W., D. G. Noller, T. B. Robey (1986), "Experiences with Analytic and Simulation Modeling for a Factory of the Future Project at IBM", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, New Jersey, 641-648.

Hoffman, S. E., M. M. Crawford, and J. R. Wilson (1983), "An Integrated Model of Drilling Vessel Operations", presented at the Winter Simulation Conference, Arlington, Virginia.

Jonatansson, E. and S. U. Randhawa (1986), "A Network Simulation Model of a Fish Processing Facility", in Simulation, July.

Lilegdon, W. R., C. H. Kimpel and D. H. Turner (1982), "Application of Simulation and Zero-One Programming for Analysis of Numerically Controlled Machining Operations in the Aerospace Industry", presented at the Winter Simulation Conference, San Diego, California.

Logendran, R. and M. P. Terrell (1986), "Program Planning and Development of a National University Teleconference Network Using Simulation", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, New Jersey, 776-785.

Lu, K. H. and L. Van Winkle (1984), "A Critical Evaluation of Some Problems Associated with Clinical Caries Trails by Computer Simulation", in Journal of Dental Research, May, 796-804.

Martin, D. L. (1983), "A Simulation-Optimization Model for Exploration and Exploitation of Exhaustible Mineral Resources", presented at the SME-AIME Annual Meeting, Atlanta, Georgia.

Martin, D. L. (1986), "Simulation Analysis of an FMS During Implementation", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, New Jersey, 628-632.

McCallum, J. N. and B. B. Nickey (1984), "Simulation Models for Logistics Managers", in Logistics Spectrum, Volume 18, No. 4, Winter.

Morris, W. D., T. A. Talay and D. G. Eide (1983), "Operations Simulation for the Design of a Future Space Transportation System", presented at the AIAA 21st Aerospace Sciences Meeting, Reno, Nevada.

Murphy, D. R., S. D. Duket and E. Sigal (1985), "Evaluating Surgical Block Schedules Using Computer Simulation", in Proceedings of the 1985 Winter Simulation Conference, D. T. Gantz, G. C. Blais, and S. L. Solomon, Eds., IEEE, Piscataway, New Jersey, 551-557.

O'Reilly, J., M. Sale, and D. Martin (1985), "The Use of Continuous/Discrete Event Models in Manufacturing", in Proceedings of the 1985 Winter Simulation Conference, D. T. Gantz, G. C. Blais, and S. L. Solomon, Eds., IEEE, Piscataway, New Jersey, 308-314.

Prestwood, W. T., "PATRIOT Air Defense Weapon System Deployment Model Using SLAM", U.S. Army Missile Command Logistics Support Analysis Office, Redstone Arsenal, Alabama.

Pritsker, A. A. B. (1982), "Applications of SLAM", IIE Transactions, March, 70-77.

Pritsker, A. A. B. (1986), Introduction to Simulation and SLAM II, Third Edition, Halsted Press, New York, New York.

Pritsker, A. A. B., C. E. Sigal, and R. D. Hammesfahr (1989), SLAM II Network Models for Decision Support, Prentice-Hall, Englewood Cliffs, New Jersey.

Pritsker Corporation (1990), SLAM II Quick Reference Manual, Pritsker Corporation, West Lafayette, Indiana.

Ratcliffe, L. L., B. Vinod and F. T. Sparrow (1984), "Optimal Prepositioning of Empty Freight Cars", in Simulation, June.

Sale, M. and C. W. Stein (1987), "Modeling AGV Systems Using Network Constructs", presented at the Winter Simulation Conference, Atlanta, Georgia.

Shevell, S., J. Buzacott, and M. Magazine (1986), "Simulation and Analysis of a Circuit Board Manufacturing Facility", in Proceedings of the 1986 Winter Simulation Conference, J. R. Wilson, J. O. Henriksen, and S. D. Roberts, Eds., IEEE, Piscataway, New Jersey, 686-693.

Standridge, C. R., and J. R. Phillips (1983), "Using SLAM and SDL to Assess Space Shuttle Experiments", in Simulation, July, 25-35.

Standridge, C. R. and A. A. B. Pritsker (1987), "TESS: The Extended Simulation System", Halsted Press, New York, New York.

Tsui, J. S. (1985), "Managing Critical and Expensive Equipment Spares Through Simulation", presented at the AIIE Conference, Los Angeles, California.

Ueno, N., Y. Nakagawa, Y. Okuno, S. Morito, "Steel Product Transportation and Storage Simulation: A Combined Simulation/Optimization Approach", in Proceedings of the 1988 Winter Simulation Conference, M. A. Abrams, P. L. Haigh, and J. C. Comfort, Eds., IEEE, Piscataway, New Jersey, 678-683.

Wilson, J. R., D. K. Vaughan, E. Naylor, and R. G. Voss (1982), "Analaysis of Space Shuttle Ground Operations", in Simulation, June, 187-203.

Author biography

Jean J. O'Reilly is Director of Training and Support at Pritsker Corporation in West Lafayette, Indiana. She holds a Bachelor of Arts degree in Mathematics from St. Mary's College, Notre Dame, Indiana and an M.S. in Applied Mathematics from Purdue University. Since joining Pritsker Corporation in 1978, Ms. O'Reilly has been involved in software development and in applying SLAM II in various consulting projects. In her current position, she is responsible for technical support and training for all Pritsker Corporation products.

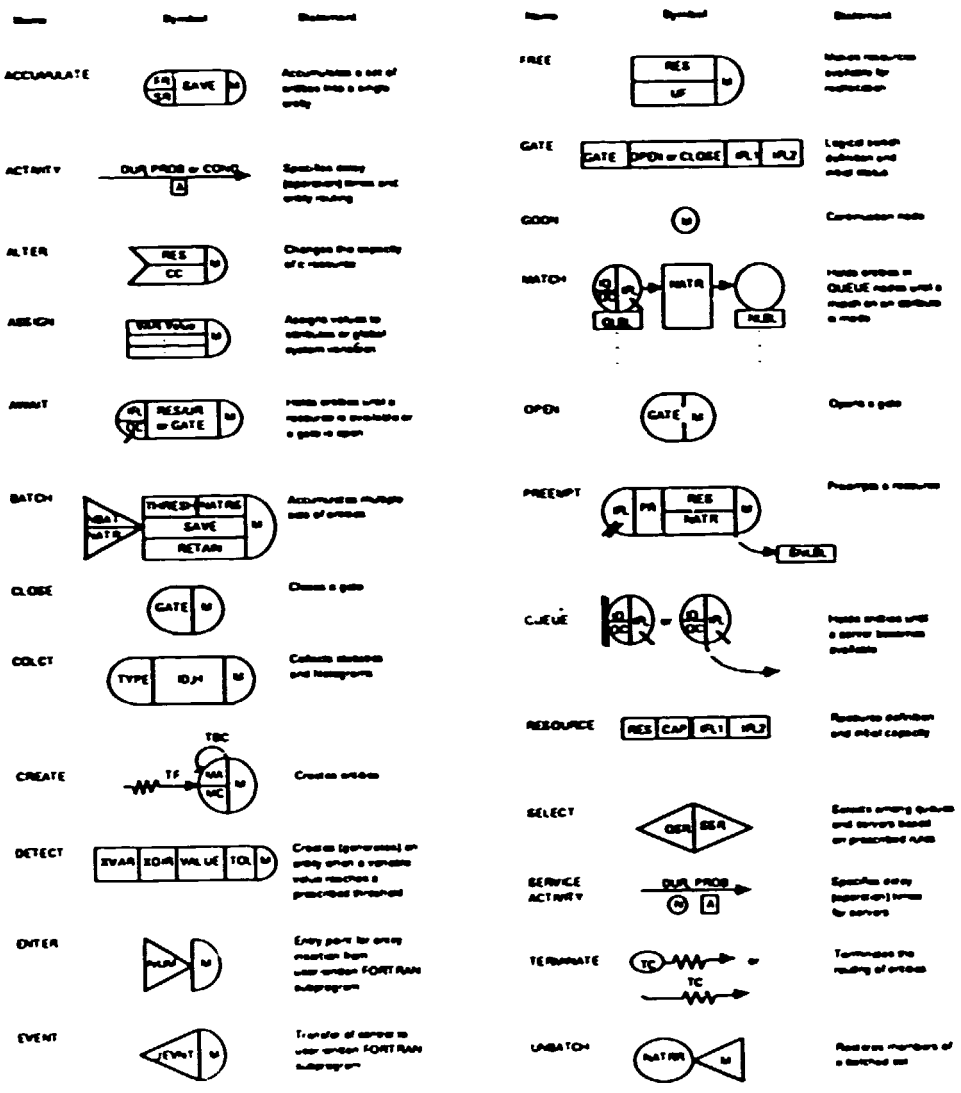


Figure 1. SLAM II Network Symbols

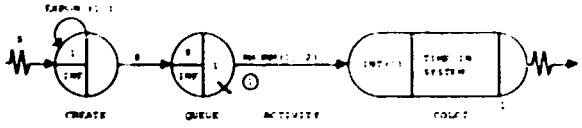


Figure 2. A Single-Server Queuing Example

```
CREATE,EXPON(1.),0.,1;          GENERATE ARRIVALS
QUEUE(1);                       WAIT FOR SERVICE
ACTIVITY(1),RNORM(1.,2);        PROCESS
COLCT,INT(1),TIME IN SYSTEM;    COLLECT STATISTICS
ENDNETWORK
```

Figure 3. Example Model Input

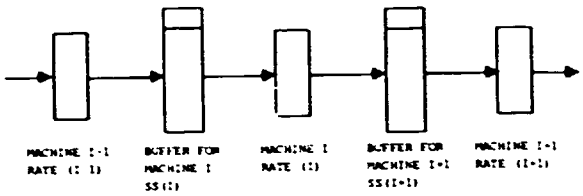


Figure 4. Modeling with Continuous Variables

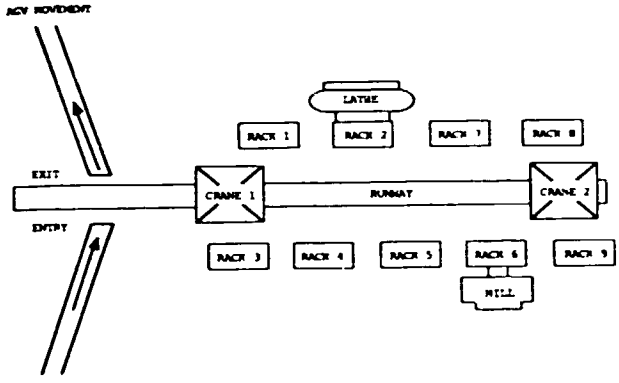


Figure 5. Schematic Diagram of an ASRS System

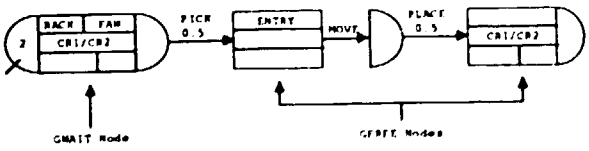


Figure 6. Movement from Entry Point to Storage

9. TUTORIAL: SCHEDULING MANUFACTURING SYSTEMS WITH FACTOR

David Krahl

Pritsker Corporation
1305 Cumberland Avenue
West Lafayette, Indiana 47906

1. Abstract

This tutorial covers the basic concepts of FACTOR version 5.0 as applied to scheduling a production facility. Topics include: the FACTOR 5.0 modelling constructs, integration with existing production data, and the use of FACTOR for schedule creation and adjustment.

2. Scheduling and simulation

Scheduling in most manufacturing facilities is currently performed assuming no limits on the capacity of the manufacturing personnel and equipment. Because of the ease of modelling limited resources, simulation is able to provide more accurate and detailed schedules and performance estimates than traditional methods. Traditional simulation tools, however, have not been designed to handle the specific requirements of a scheduling application. These include: the ability to load the current manufacturing system status including the actual orders being processed, providing accurate, detailed equipment, material, and personnel schedules, and providing the required simulation results before they become out of date on the manufacturing floor. In addition to these requirements, scheduling software must be able to interface directly with existing production control systems to allow automated data transfer, both in to and out of the scheduling tool. Finally, results must be presented in a manner which is easily understandable by shop floor personnel who most likely will not be familiar with traditional simulation terms analysis.

3. Factor-scheduling decision support utilizing simulation

FACTOR is a decision support software system which, through the use of a simulation kernel, is able to generate detailed finite capacity schedules, accurate capacity planning information, and on-line schedule adjustment. FACTOR is specifically designed to meet the needs of manufacturing production planning. In addition, sufficient flexibility has been incorporated to ensure that the required level of detail can be achieved.

While FACTOR is not a simulation language, a model is built by combining basic modelling components in a way which duplicates the characteristics of the actual system. These model components are stored in a fast access database before the start of the simulation of the production system. This information can be either loaded manually through standardized or customized screen oriented editors or utilities. Input error checking and on-line help are available both for the standard system and any user customizable options.

FACTOR output for simulated alternatives is also stored in the database. This allows for transfer of the required information to external manufacturing systems through a user customizable export utility. This output may be generated by the standard FACTOR report generator or tailored to the specific needs of the application and viewed on a computer terminal with a full screen review function.

The FACTOR simulator is coded entirely in the C programming language and uses advanced techniques for rapid simulation execution and schedule generation. Currently FACTOR 5.0 is available on the IBM AS/400 with the schedule adjustment functionality residing on OS/2. FACTOR 4.0 is available on VMS, HP/UX, OS/2, and VM.

4. Factor modelling components

FACTOR 5.0 provides an extensive set of standard modelling components for use in building models of production systems. The major components include:

- Order characteristics;
- Shop floor status;
- Production calendar;
- Shift schedules;
- Resources;
- Functional resource groupings;
- Resource maintenance;
- Parts;
- Materials;
- Process plans.

Resources are either classified as single capacity resources (FACTOR resources) or multiple capacity resources (pools). FACTOR resources are generally used to model personnel and manufacturing equipment which can be on a shift, subject to maintenance, or for which schedule information is desired. Multiple units of a resource are modelled by making a FACTOR resource a member of a resource group. A resource requirement for an operation is thus listed as the resource group rather than any of its members. A variety of resource group member selection rules including user customizable rules are available. Pools are used to model resources such as a WIP area for which no schedule, maintenance, or shift is required. FACTOR includes a number of the most commonly used scheduling rules and a convenient interface for installing user defined logic.

FACTOR process plans provide details on how and in what order operations are performed on a part. A process plan consists of a sequence of jobsteps with provisions of a standard and alternative routing at each point. Each jobstep can require one or more resources, resource group members or pools. The processing duration may be defined at the jobstep as fixed for the entire

load, pre-piece or calculated in a user function. The major FACTOR jobsteps include:

- Operation;
- Setup/operation;
- Setup;
- Move;
- Move between;
- Assemble;
- Produce;
- Inspect;
- Batch;
- Add to material;
- Remove from material;
- Accumulate and split.

In addition to these jobstep types, the modeller may create specialized modelling components of their own design. User defined components include: jobsteps, resource sequencing rules, jobstep durations, resource group member selection, and alternative jobstep selection rules.

5. Data integration capabilities

One of the major factors in the success of a scheduling system is the integration of the scheduling software with existing production data systems. Accurate schedule generation depends upon accurate production system objectives and status at the beginning of the simulation. The ability to import information from other sources with speed and ease is critical to the success of a scheduling product.

Although it is possible to enter all of the required data to generate a schedule manually, to achieve the required level of automation, most data will be placed into the FACTOR database through the use of transfer programs written with tools such as SQL, RPG, or C. If the external manufacturing systems reside on the AS/400 along with FACTOR 5.0, the data transfer operation requires that the necessary information be merely copied into the FACTOR 5.0 data base records.

Often the ultimate end user of a scheduling package will be a person with little or no knowledge of the actual inner workings of the software that generated the schedule. It is critical that the information provided by the scheduling package be in a form that is in the language of the person interpreting the schedule on the factory floor. All of the functions must be easily accessible without being burdened with modelling details or data fields for which the shop floor personnel have no control.

FACTOR provides two standard interfaces, the scheduler's interface and the modeller's interface. The scheduler's interface gives the shop scheduler access to information necessary for the creation and evaluation of various scheduling alternatives. In addition to the functionality of the scheduler's interface, the modeller's interface gives the FACTOR modeller access to information necessary for the creation and maintenance of the scheduling model. Both of these interfaces are tailorable to the user's environment. This feature is especially important to the scheduler as it allows the FACTOR information to be presented in a manner consistent with the application environment.

The interface tailoring option also provides the user with the ability to create a totally new interface for either the modeller or the scheduler. The FACTOR software user has complete control over screen content, screen organization, help messages, input checking, and the commands which will be executed for a selected option. It is possible to integrate functions defined outside of FACTOR such as the initiation of a data transfer function. This functionality provides a single consistent interface for the entire scheduling operation.

6. Scheduling and schedule adjustment with FACTOR

In practice, FACTOR is used to schedule operations on a regular scheduling interval and to handle unexpected events. At the start of the scheduling interval (shift, day, week ...) status information is transferred into the FACTOR database. The scheduler executes the simulation and reviews a summary of the performance of the schedule. These reports allow the scheduler to detect potential scheduling problems and adjust the parameters of the FACTOR model. The new model is then executed and the two alternatives are compared. This process is repeated until a satisfactory schedule is created. Once an alternative has been accepted, a detailed schedule is generated for the components of interest (equipment, personnel, materials ...). This information can be distributed to the operators or automated cell controllers.

Often, one or more unforeseen events may invalidate the current schedule. These events include a machine failure, the arrival of a rush order, or a missed delivery date of a supplier. To react to these situations the FACTOR 5.0 Schedule Management Module (SMM) could be used to interactively adjust the schedule to meet the new system conditions.

The SMM is a graphical scheduling tool which provides a convenient mechanism to review and quickly adjust an existing FACTOR 5.0 schedule. The schedules presented by SMM are in the form of interactive GANTT charts (see figure 1). The three functions provided by SMM are: schedule viewing, schedule adjustment, and schedule transfer. Schedule viewing displays the current schedule for orders, jobs, and resources allowing the scheduler to quickly review critical information. Schedule adjustment allows the scheduler to react to unexpected changes in the shop floor status by graphically editing the current schedule. Schedule transfer allows the scheduler to transfer the schedule information to and from a machine running the OS/2 operating system where SMM resides to the FACTOR 5.0 database on an AS/400.

The function of FACTOR in the scheduling environment is thus both a tool to generate feasible and achievable schedules as a decision support system for rapid "what if?" analysis of scheduling alternatives. The scheduler is with the capability to completely and accurately determine the outcome of a scheduling decision, adjust the schedule to meet the constantly changing shop floor status. This capability is a

necessity when scheduling the highly complex production systems in use today.

7. The FACTOR tutorial

The tutorial at the Winter Simulation Conference will provide details about modelling and scheduling with FACTOR. The presentation will include a series of FACTOR screens, details about modelling components, example output reports and a demonstration of the Schedule Management Module. The modelling process will be discussed in detail as will the implementation and integration of the model, and the use of the model for scheduling.

8. Additional reading

Grant, Floyd H. (1987), Scheduling and Loading Techniques. Production and Inventory Control Handbook, Second Edition, Green, J. H., Ed., American Production and Inventory Control Society.

Grant, Floyd H. (1986), Production Scheduling Using Simulation Technology, Advanced Manufacturing Systems Conference, IFC (Conferenced) Limited, 129-138.

FACTOR Implementation Guide (1989), Pritsker Corporation, West Lafayette, Indiana.

FACTOR Site Specific Tailoring (1989), Pritsker Corporation, West Lafayette, Indiana

FACTOR User Interface Tailoring (1989), Pritsker Corporation, West Lafayette, Indiana.

FACTOR Information Transfer (1989), Pritsker Corporation, West Lafayette, Indiana.

FACTOR Output Analysis System (1989), Pritsker Corporation, West Lafayette, Indiana.

MacFarland, D. G. and F. H. Grant (1987), "Shop Floor Scheduling and Control using Simulation Technology", Shop Control 1987, Cincinnati, Ohio.

Author biography

David Krahl is a Consultant in the Training and Support group at Pritsker Corporation. He received a Bachelor of Science Degree in Industrial Engineering from the Rochester Institute of Technology. Since joining Pritsker in 1986, he has performed technical support and taught classes in Pritsker software products, developed simulation application for the consumer products, automotive, and aerospace industries, and worked in software design and development.

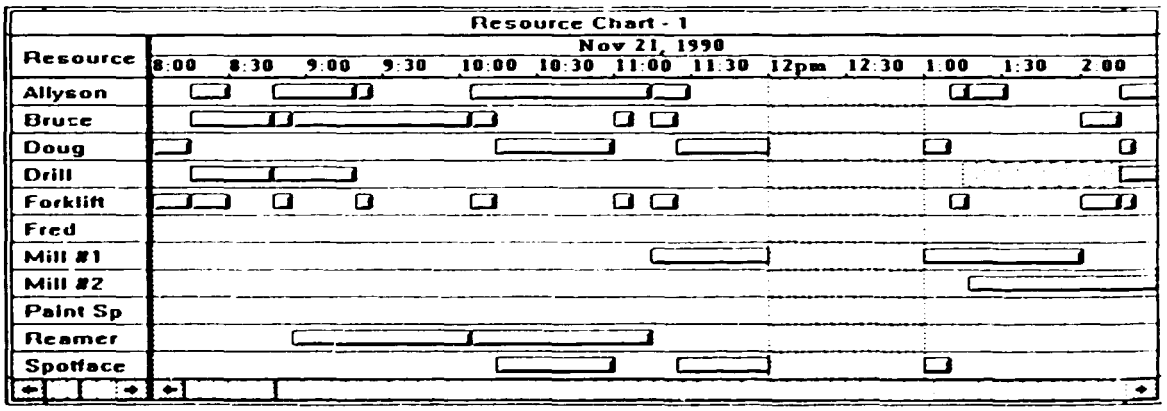


Figure 1. Schedule Management Module Resource Chart

10. PROMODEL TUTORIAL

Charles R. Harrell, Ph.D.
Ken Tumay

Production Modeling Corporation International
1875 South State, Suite 3400
Orem, Utah 84058

Abstract

ProModel combines the flexibility of general purpose simulation languages and the easy-to-use features of manufacturing simulators. This tutorial presents ProModel's design philosophy, describes its modeling elements, and illustrates these elements with an example.

1. Design philosophy

In this section, we describe the design criteria used in developing ProModel.

ProModel appeals to a broad user base. ProModel is designed to be used by novice simulation users as well as simulation experts. Industrial and manufacturing engineers have neither the time nor the interest to do programming and yet they have the need for a modelling tool that is powerful enough to simulate a wide range of production systems. ProModel is an easy-to-use and convenient tool for engineers and managers who are interested in quick results. Because of its ease of use, it is also attractive to professors in engineering or business programs who are interested in teaching modelling and analysis concepts rather than teaching programming.

When simulating complex systems that require extensive analysis, usually a simulation expert with programming skills is involved in the modelling activity. In such situations, total modelling flexibility can only be achieved through additional programming. To satisfy this need, ProModel offers complete programming capability in a C or PASCAL type language which can be conveniently accessed without exiting from the program. In this respect, ProModel is powerful and convenient for systems analysts and simulation experts who are interested in ultimate flexibility.

Model development is completely graphical and object-oriented. To the extent possible, all input is provided graphically and information is grouped by objects for quick and intuitive access. This data input approach minimizes the learning curve for beginners and maximizes the efficiency for modifying large and complex models.

Powerful manufacturing constructs minimize model development time. Such typical manufacturing constructs as AGVs, conveyors, cranes, robots are not available in most general purpose languages. On the other hand, those simulators that claim to offer these constructs have rudimentary capabilities that oversimplify the actual hardware characteristics. ProModel offers realistic and flexible constructs for modeling complex manufacturing systems quickly.

Object oriented programming using C++ creates robust and portable software. One of the shortcomings of simulation software has been the difficulty in porting models across a variety of operating systems or hardware platforms. The main reason for these shortcomings is the underlying languages used such as FORTRAN. ProModel takes advantage of the portability and object orientation provided by C++. Furthermore, C++ produces very efficient code that is less prone to bugs and easy to add new features.

The latest advancements in operating systems technology are utilized. ProModel takes advantage of the state-of-the-art memory management techniques, synchronized windowing and dynamic data exchange capabilities offered by WINDOWS and OS/2. Fully compliant with CUA (Common User Access) standards, ProModel allows multiple applications to run concurrently.

Model size is limited only by memory and execution speed is increadibly fast. Model size and execution speed have long been two significant concerns for simulation users on microcomputers. Although those simulation tools running under WINDOWS and OS/2 have eliminated the model size limits, they have done so by sacrificing model size for speed. Unlike other simulation software products, ProModel provides unlimited model size while offering extremely fast execution speed.

Graphics are realistic and easy-to-develop. Realistic looking animation helps simulation to become a powerful communication vehicle between engineers and managers. However, most engineers have neither the time to create 3-D graphics nor the easy access to special graphics terminals. ProModel offers a colourful and powerful 2-D graphics editor with scaling, rotating, etc. capabilities on standard hardware. CAD drawings (e.g. AutoCAD), scanned pictures and drawings can be imported into ProModel for animation development.

Animation development is integrated with model definition. A major drawback of many simulation software products is that animation development is independent from simulation model development. This makes it time consuming and inconvenient for engineers to use animation as a validation/verification tool. ProModel integrates system definition and animation development into one function. While defining routing locations, conveyors, AGV paths, etc., you essentially develop the animation layout. The layout screen is a virtual screen which can be scaled to an actual factory layout.

Simulation results are easy to generate, meaningful and graphical. Most simulation software products require special commands to generate statistics that are difficult to interpret for non-simulationists. ProModel allows quick and convenient selection of reports and provides automatic tabular and graphical reports on all system performance measures. The reports provide meaningful detailed statistics; thus

eliminating special commands to generate useful information.

ProModel runs with standard hardware. Most engineers, managers, and professors have easy access to IBM or compatible computers with VGA graphics capabilities. ProModel does not require any special graphics card, special monitors, or a math coprocessor. This makes it convenient and cost effective for companies and academic institutions that have standard micro-computers. ProModel is also available on LANs.

2. Modelling elements

In ProModel, a model defines a production or service system which consists of the items being processed, one or more processing locations, any number of auxiliary resources, routing and operation logic and a production plan or schedule. Modelling elements in ProModel are the constructs used to represent the physical and logical components of the system being modelled.

Physical elements of the system such as parts and resources may be referenced either graphically or by name. Names may be any word or combination of words up to 32,000 characters long. Following is a brief description of each of these elements.

2.1 Parts or entities

Parts or entities refer to the items being processed in the system. These include raw materials, piece parts, assemblies, loads, WIP, finished products, etc. Entities of the same type or of different types may be consolidated into a single entity, separated into two or more additional entities or converted to one or more new entity types.

Entities may be assigned attributes that can be tested in making decisions or for gathering specialized statistics.

2.2 Routing locations

Routing locations are fixed places in the system (e.g. machines, queues, storage areas, work stations, etc.) to where parts or entities are routed for processing, storage or simply to make some decision about further routing. Route locations may be either single unit locations (e.g. a single machine) or multi-unit locations (e.g. a group of similar machines performing identical operations).

Routing locations may have a capacity greater than one and may have periodic downtimes as a function of clock time (e.g. shift changes), usage time (e.g. tool wear), usage frequency (e.g. change a dispenser after every n cycles), change of material (e.g. machine set-up) or based on some user defined condition.

2.3 Resources

A resource may be a person, tooling, vehicle or other device that is used to transport material between routing locations, or to perform an operation on material at a location, or to perform maintenance.

Resources may be any one of the following three types:

General resources which are immediately accessible when not in use and have no special operating characteristics,

Mobile resources which have special operating characteristics to define their movement between tasks,

Robots which have an end of arm tool or gripper,

Cranes (bridge or gantry) which are mounted overhead and move along multiple axes.

2.4 Conveyors

A conveyor is a continuous movement device along which entities are conveyed. Conveyors are accumulating or non-accumulating and may be segmented or non-segmented. Conveyors have operating characteristics such as speed and material spacing and may also have downtime characteristics. Conveyors may be configured with transfers, recirculation loops, sortation, accumulation and distribution capabilities. Bi-directional conveyors and complex conveyor networks can be modelled.

2.5 Variables

ProModel provides numerous variables for decision making and statistical reporting. Variables are of two major types: (1) system or state variables such as the clock time or available capacity of a resource, and (2) user-defined variables including gates and multi-dimensional arrays.

2.6 Functions and distributions

In addition to variables, ProModel has numerous built-in functions and distributions including discrete and continuous empirically defined distributions. Built-in distributions include exponential, normal, uniform, triangular, beta, gamma, erlang, weibull and lognormal.

2.7 Logic and action statements

To perform special testing and provide specific instructions within a model, ProModel enables the user to enter logic and action statements. While some statements are related specifically to entities and resources, other statements are general programming statements providing the complete flexibility of a programming language including if-then statements, switch or case statements, loops, complex boolean expressions and file I/O statements. Statement nesting and even sub-routines are supported.

3. Using ProModel

Much of the ease in using ProModel comes from the simple and straightforward way in which models are defined and results are obtained. A model is defined the same way in which an engineer would naturally describe a production system using the same familiar terminology. A unique "Automatic Model Build" option is available for walking the user through the complete model building process. On-line, context sensitive help screens provide useful hints for data input.

In this section, we will define a problem, step through the model building process, describe the simulation run and present the output results.

3.1 Problem definition

To illustrate how ProModel works, we define a Flexible Manufacturing Cell with six operation stations, a load station, an unload station and an AGV system (see figure 1 for the ProModel layout of the model). Pallets are scheduled into an Input queue based on an exponential distribution. Each pallet contains five blades. From the input queue, individual blades are moved to a Load station where they are loaded onto a fixture by an operator (Operator 1). Once the fixture is loaded, an AGV is requested to move it to Machine A or Machine B (Station 1). From either Machine A or B, a fixture is sent to either Machine C or Machine D (Station 2). After processing on Machine C or D, the fixture is sent to an on-line Inspection station. If the blade passes inspection, the fixture is sent to the Unload station. If the blade fails, it will be sent to a Rework station, repaired and then sent to Unload. The probability of having a reject is .045. At the Unload station, the blade is taken off the fixture by an operator (Operator 2). The empty fixture is sent to the Fixture queue waiting to be loaded with another blade. The system is modelled with five fixtures and three AGVs.

3.2 Building the model

To build the model, first the routing locations such as the load stations and machines are defined and placed on the layout. Operations performed for each part type at each location is then entered. After the routing locations and operations have been defined, auxiliary resources including the operators and AGVs are defined. With all of the locations and resources identified, the parts or entities and their associated routing through the system can be defined. Finally, the scheduling of parts into the system and simulation parameters are defined.

3.3 Defining the model locations

Locations are defined by choosing the Location module from the Module Definition menu and entering a name, capacity and number of units for each of the locations in the system. Operations performed at each location are then defined.

To place a location on the screen layout, a library or user drawn graphic is selected (see figure 2) and dropped on the screen layout. These graphics may be moved, sized and even rotated. Any number of additional graphics (e.g. labels, counters, etc.) may be defined for the location.

This particular model has the following location definitions:

Location	Capacity	Units
Fixt Que	10	1
Input Que	20	1
Load	1	1
Station A	1	2
Station B	1	2
Inspect	1	1
Rework	1	1
Repair	1	1
Unload	1	1

Operations at a location are defined based on part type. The following operations are defined for this model:

Part	Location	Operation or action
Fixture	Fixt Que	0
Pallet	Input Que	Split 5 blade
Fixture	Load	Join 1 blade Use Op 1 for 1
Fixture	Station A	2
Fixture	Station B	3
Fixture	Station C	1
Fixture	Rework	U(3, 1.5)
Fixture	Unload	Use Op 2 for U(1, .5)
Fixture	Inspect	N(1.5, .25)

3.4 Defining auxiliary resources

Additional resources required for processing or material handling are defined through choosing the Resource module and entering the name and any operating characteristics for each resource.

The operators in this model are considered to be general resources that, at least for modelling purposes, are assumed to be immediately accessible when available. For general resources, only the number of units and screen position need be defined. This model uses two general resources (Operator 1 and Operator 2) consisting of 1 unit each.

The AGV is a mobile resource which requires that a path network be defined with speeds and distances. Location interface positions and other control points are defined as well as any work search priorities. Idle vehicle deployment may also be specified. The AGV has a path layout that is defined graphically with distances either automatically supplied according to scale or manually entered. The operation parameters for the AGV are as follows:

Speed ft(m)	Pickup (sec)	Deposit (sec)	Search rule	Acc., Dec. (fpss)
120	4	4	OLDEST	3

3.5 Defining parts and associated routings

Parts are defined by invoking the Part or Entity module and entering a name for each part type. A predefined or user-defined graphic may also be selected to represent the part. For each part type, a routing is defined by selecting a "from" location and one or more alternative "to" locations. Locations are selected either by entering the location name, selecting the location from a help menu or by simply clicking the mouse on the location. Any location selection rules are entered as well as any move time or movement resource used to make the move.

For our demonstration model, there are three part types defined: fixture, pallet and blade.

The routing for fixture is defined as follows:

From	To	Rule	Resource	Time
Fixt Que	Load	0	0	1.5
Load	Station A	0	AGV	0
Station A	Station B	0	AGV	0
Station B	Inspect	0	AGV	0
Inspect	Rework	95.5%	0	.5
Inspect	Unload	4.5%	AGV	0
Rework	Unload	0	AGV	0
Unload	Fixt Que	0	0	.5

Note that no time is specified for movement of the AGV since it is a mobile resource whose time is determined by the speeds and distances defined in the Path Logic module.

The routing for blade is as follows:

From	To	Rule	Resource	Time
Input Que	Load	JOIN	0	:30

Note that there is no routing specified for pallet since it changes name after splitting.

3.6 Scheduling part arrivals

With all of the locations, resources and part routings defined, the only thing left is to give the system work to perform. This is done by defining the part arrivals in the Arrivals module. Each part arrival is defined by entering a part name or selecting a part from the parts menu. The location where the parts are to arrive is then selected by name entry or clicking on the location. The quantity per arrival and frequency of arrivals are also defined.

3.7 Running the simulation

The length of the run and other simulation options are defined in the General Information module. The simulation can be run with or without animation. The animation may even be only temporarily disabled to speed ahead. Additionally, the state of the system can be saved at any point in time which can be recreated instantly without having to run the entire simulation again.

3.8 Analysing the results

The results of the simulation run contain resource utilization, queueing, and throughput statistics. More detailed statistics for producing histograms and time series charts can be collected.

4. Conclusions

Until recently, manufacturing companies have not fully benefited from simulation in making continuous improvements because of the time, programming expertise, and cost involved in getting useful results. ProModel is designed for manufacturing companies to fully achieve the benefits of simulation technology at an affordable price. ProModel is directed toward making simulation a standard tool in the hands of engineers, managers and systems analysts just as spreadsheet software is in the hands of accountants and financial analysts.

References

Harrel, C. R. 1989. PROMOD (PROduction MODeler) for IBM PC's. In Supplementary Proceedings of the SCS Multiconference, eds. S. Spenser and G. Richardson, 65–70. San Diego, California.

Harrell C. R. and K. Tumay. 1990. ProModelPC Tutorial. In Proceedings of the Winter Simulation Conference, eds. O. Balci, R. Sadowski and R. Nance, 128–131. New Orleans, Louisiana.

Harrell C. R. 1990. Trends in Manufacturing Simulation. In Proceeding of Autofact Conference, eds. A. Adlard, 21–31. Detroit, Michigan.

Tumay, K. 1989. Opportunities and Challenges for Busy Engineers in Manufacturing Simulation (Panel). In Proceedings of the Winter Simulation Conference, eds. E. MacNair, K. Musselman and P. Heidelberger, 859–864. Washington D.C.

Production Modeling Corporation of Utah. 1991. ProModelPC User's Manual.

Author biographies

Charles R. Harrell, Ph.D., is the founder and President of Production Modeling Corporation International. Charles has received his B.S. in Manufacturing Engineering Technology from Brigham Young University, M.S. in Industrial Engineering from University of Utah and Ph.D. in Manufacturing Engineering from the Technical University of Denmark.

Ken Tumay received his B.S. and M.S. degrees in Industrial Engineering from Arizona State University. Prior to joining PMCI, Ken worked for the Confacs Group and CACI Products Company. He is a member of IIE and SME.

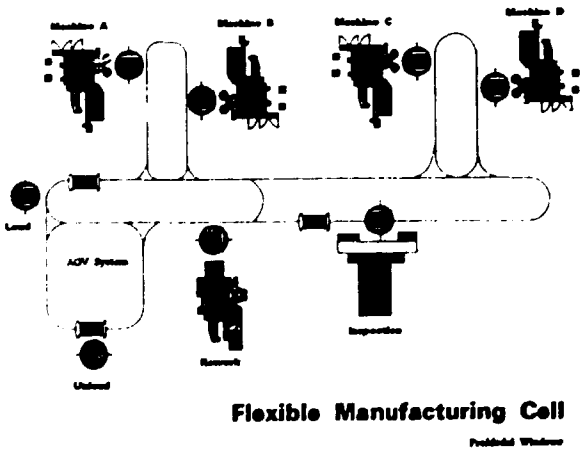


Figure 1. A ProModel Layout Screen

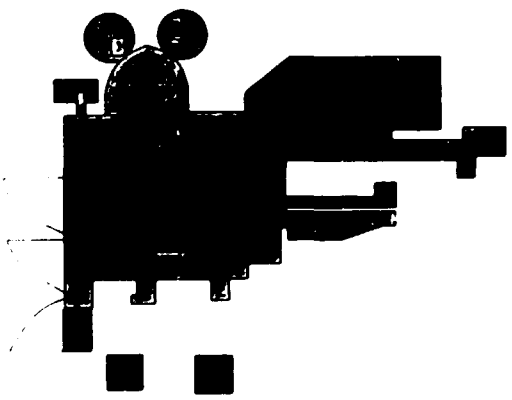


Figure 2. A detailed view of an icon

11. PASION TUTORIAL

Stanislaw Raczynski

Universidad Panamericana
Augusto Rodin 498
03910 Mexico City, Mexico

Abstract

PASION is a process- and event-oriented simulation language designed for PASCAL users. The language has a two-level (process/event) structure and permits the use of all the Pascal structures. It also offers the main features of object-oriented programming. PASION provides necessary facilities to handle sequences of random events, queues and quasi-parallel processes, both discrete and continuous.

1. Why a new language?

A look at the annals of simulation software development could result in the impression that we have enough (or, rather, too many) simulation languages and packages. The idea of creating a new one might appear to be crazy. However, looking at the existing simulation software and at the recent tendencies in programming, it can be seen that the existing simulation software becomes somewhat obsolete. The new simulation software should be object-oriented not only because almost all new software is object-oriented (see Schmucker, 1986, for a review on object-oriented programming). The simulation software must have this orientation simply because the real world is object-oriented. Second, it is not a proper way to develop good simulation tools modifying or extending languages which are 20 or even 30 years old. In my didactic work I have been looking for a well structured and easy-to-teach simulation language. It seems that the most complete one is Simula. However, it is somewhat difficult to teach Simula quickly, for its relation to Algol. Similar difficulties appear when using Modula-2. Shortly speaking, the new language should be object-oriented, and should offer the following features: clear process/event structure, efficient clock mechanism, combined continuous/discrete modelling and inheritance. It also should have an evironmment which supports graphics, interactive simulation and auxiliary modules (program generators) for queuing and continuous models.

2. The language

Let us recall some basic concepts of process-oriented simulation. To describe a sequence of events we must specify event operations and describe both dependence of each event on the model time, and the interactions between the events. A process-oriented language offers something more. Namely, it defines a structure within the set of events by introducing different processes. By the process we mean a generic program segment which declares a specific object type. This declaration describes the properties of objects which can execute events in relation to the model time. According to this declaration the corresponding objects can be created at run-time. This approach makes the simulation object-oriented. Recall that an object-oriented language should support information hiding, data abstraction, dynamic binding and inheritance (see Dahl and Nygaard 1967, Koehler and Patterson 1986a, 1986b; Pascoe 1986; Schmucker 1986).

PASION does not fully implement all these concepts. It supports dynamic creation of objects, data hiding and, to some extent, data abstraction by the mechanism of predefined processes and inheritance. The following example shows a PASION program where two objects activate each other.

```
PROGRAM TRIGGER;
REF A,B:X;
{A and B point to objects of type X}

PROCESS X,2;  A process type
ATR N:STRING[6];

EVENT ONE; {ONE is an event}
WRITELN  'Active object: ',N ;
IF THIS=A THEN B.ONE:=TIME+1.0
ELSE A.ONE:=TIME+1.0 ENDEV;

START {Main program}
NEWPR A;A.N:='FRED';
NEWPR B;B.N:='ANDREW';
A.ONE:=TIME;
{The object A starts immediately.
B waits.}
${This terminates the program.}
```

This is a complete PASION program. The reference variables A and B are used to refer to the two objects of type X. The instruction A.ONE:=TIME+1.0 schedules the event ONE of the object A to be executed at TIME+1.0, where TIME is the model time. THIS is a reserved word which can be used to refer to the object from within its scope. Thus, THIS=A is true when the object is referred by A and false for the object referred by B. It can be seen that the two objects activate each other. The output from this simple program is as follows:

```
Active object:  FRED
Active object:  ANDREW
Active object:  FRED
Active object:  ANDREW
Active object:  FRED ... etc.
```

It is a very simple example. In practice, the simulation programs can have up to 50 process declarations (object types) and up to 400 events in each process. The number of objects generated at run time depends of the complexity of the objects. Models with more than 2,000 objects were successfully run on the IBM XT.

3. Correspondence between models and PASION programs

According to the commonly used simulation terminology (see Zeigler, 1976) a simulation model is composed by its components (e.g. clients in a shop). The state of each component is described by the corresponding set of descriptive variables and its activities are given by the rules of interaction between the components. Experimental frames define the actually used set of descriptive variables and determine the complexity of the model. The PASION language has all these basic modelling elements. Model components are objects,

component specificiation is given by the process declaration, descriptive variables are process attributes and the component activities are events. Experimental frames can be expanded using the mechanism of inheritance. Inheritance enables programmers to create classes and therefore objects that are specializations of other objects. This enables the programmer to create complex models by using code created and tested before. Inheritance in PASION can be applied using prefixed process declarations. Let PA be the name of an existing process. Suppose that we wish to create a new one, say PB, having all the properties of the process PA (this means all its attributes and events). This can be done using the name PA/PB instead of PB in the heading of the process declaration. While processing such declaration, the translator looks for the process PA (the parent process) and inserts all the attribute declarations and event descriptions from PA into the new process PB (derived process). Parent processes can reside in separate files, or be placed in the same source file. Thus, the user can prepare and store some useful source "capsules" and use them while creating new processes. During the creation of the new (derived) process some of the names used in the parent process can be changed. This includes variable or type identifiers.

4. Some examples

Let us consider a simple example of object-oriented simulation in PASION. To simulate the growth of a plant it is sufficient to describe the behaviour of one "cell" of the plant. It can be "branch element" which can generate one or more other branch elements which grow upwards, with random inclination. The branch element can also increase its thickness to "support" more branches. The program describes one branch element with two events: "generating new branch" and "to get fatter". The main program generates one initial core element which generates the branches (other objects of the same type), which, in turn, generate other branches etc. It is easy to show this process on the screen, as indicated in figure 1.

Observe that this simulation is not only the generation of the image of the plant. Each "branch element" of the plant is "alive", being an active object of the model and its behaviour can be modified in order to experiment with the model. As an example of another object-oriented simulation consider a two-dimensional heat distribution problem. Suppose that the heat propagates in a rectangular plane section. Let us discretize this region replacing it with a uniform red of points, without defining any time-discretization. Each point is an object in the simulation program. The attributes of an object are its coordinates, its temperature and the proper heat. The only activity of each object is to adjust its temperature according to the temperatures of the four nearest points and according to the heat conductivity of the material between the points (not necessarily the same for different points). The fixed boundary conditions in this model can be defined by fixing the temperature for some objects (disactivate them). The "free boundary condition" for the points at the boundary of the region consist in the fact that these points have only three and not four neighbours. The activities of the objects are programmed to be executed in random time intervals, so that no fixed time-step exists in the model. The simulation consists in generating the points and activate them. The steady state reached by the program gives the solution to the simulated heat distribution problem. The stability of such algorithm depends on some additional model parameters, not discussed here. Figure 2 shows a solution where the region consists of 400 points (20 x 20 net). The point (13,13) is a heat source with constant temperature and the point (3,3) is an ideal heat sink. The boundaries of the region are isolated from the environment (free boundary conditions). The figure shows the final steady-state situation. The dynamics of the simulated heat propagation is given by the evolution of this plot in the model time.

Simulation of such kind of the distributed-parameter systems is rather slow and not so efficient as the solution of the corresponding partial differential equation. On the other hand, observe that the object-oriented model does not involve any differential equation at all. Consequently, we are not restricted by regularity assumptions which are rather strong when the partial equations are considered. With little modifications the model can describe any strongly "irregular" system, when, for example, the properties of the material are discontinuous functions of the temperature or when two or more similar non-continuous processes interact with each other as occurs in alloy solidification problems, quite difficult to simulate using differential equations.

5. PASION environment

There are few programming languages which can be effectively used without an appropriate environment. PASION is equipped with the Minimal PASION Programming Environment (MPPE) which consists of a library of predefined processes and other modules. It supports interactive simulation, graphics, statistical analyses of the results, continuous dynamic models and queuing models. The core of MPPE consists of the library of PASION predefined processes. These are generic program segments which generate process declarations (not objects). Predefined processes are written in PASION extended by a simple "meta-language" which permits a process to have formal parameters. The user invokes a predefined process by its name and specifies the actual parameters, which are passed to the corresponding process declaration in the user program by name, before the program is translated to PASCAL. These parameters can represent not only variable names but also types, complete expressions, instructions, comments etc. The user can prepare his own predefined application-oriented processes and add them to the library.

6. Queuing model generator

PASION has a predefined type QUEUE which is a line of type FIFO, LIFO or RANDOM. It offers a number of procedures and functions to handle lines in the queuing models. Thus, the user can declare some queues and code operations on them. Other mode to simulate queuing models is provided by a module of the MPPE named Queuing Model Generator (QMG) which makes it possible to simulate systems with queues without any programming, using a graphical model description provided by the user. When applied to queuing models, QMG offers nothing

more then GPSS or SLAMII which are, perhaps, better packages to such applications. Observe, however, that in some situations it has a certain advantage. For example, the flexible manufacturing systems (FMS, see the Charles Stark Draper Lab., Inc., 1984) are controlled by quite complex algorithms and run in a complex computerized environment. Consequently, any package used to simulate FMS must be embedded in an appropriate programming environment. QMG satisfies this requirement, being related to an algorithmic, object-oriented language (Raczynski 1990).

7. Continuous model generator

This program was designed in order to facilitate simulation of dynamic continuous systems. The Continuous Model Generator (CMG) is a program generator which generates source PASION and/or PASCAL code, according to the model specificiations given by the user, mainly in graphical form. The CMG output is created in the form of a PASION process declaration which can be inserted into any PASION (continuous, discrete or combined) model. CMG also can generate a complete PASCAL program which can be run using a PASCAL compiler. The input to CMG is formulated in terms of graph diagram which describes the dynamics of the modelled system. By the graph diagram we mean a network composed of nodes and directed links. Nodes represent signals and links represent transfer functions. CMG permits the following types of links: Static linear, Static non-linear, Dynamic linear (given transfer function), Time delay, Sample-and- hold and Superlink (a complex dynamic system). The last link type (Superlink) permits to include whole dynamic model (specified earlier and stored in a file) to the model actually being created. This feature is useful while developing complex models, composed by submodels created and tested separately. When simulating combined (discrete/ continuous) systems it is possible to declare "state events" i.e. events which occur when some continuous state variables reach a specific level.

8. Implementation

PASION-to-PASCAL translator runs on the IBM PC and compatibles. The "6000" version is being developed for the IBM RISC RS-6000 computer. The code produced by the translator must be compiled by a PASCAL compiler. The resulting program expands dynamically while new objects appear, so that the number of objects which can run simultaneously depends on the amount of the operational memory available at the run time and or the size of data blocks (attributes) of the objects. PASION has been used in teaching simulation methods. It is important to have an easy-to-learn simulation tool which may be used to illustrate the concepts of process declarations, objects, events, inheritance, preprocessing and animation, when the students have some knowledge on structural programming in PASCAL and do not have any experience in simulation.

References

The Charles Stark Draper Lab, Inc., 1984, "Flexible Manufacturing Systems Handbook", Noyes Publications.

Dahl, O., Nygaard, 1986, "Simula - An Algol-based Simulation Language", Communications of the ACM No. 9, pp. 671-678.

Koehler T., Patterson D., 1986, "A Small Taste of Smalltalk", BYTE 11(8), August, 1986, pp. 145-159.

Koehler T., Patterson D., 1986 "A Taste of Smalltalk", W. W. Norton & Co., New York.

Pascoe G. A., 1986, "Elements of Object-oriented Programming", BYTE 11(8), August 1986, pp. 139-144.

Raczynski S., 1986, "PASION - Pascal-related simulation language for small systems", SIMULATION 46(6), June 1986, pp. 239-242.

Raczynski S., 1987, "PASION, Lenguaje para Simulacion: una Introduccion", Computer-World/ Mexico, Nos. 178, 179, 180, April-May 1987.

Raczynski S., 1988, "On a simulation experiment with a parallel algorithm for optimal control", Transaction of the Society for Computer Simulation, vol. 5, No. 1, pp. 87-97.

Raczynski S., 1988, "Process hierarchy and inheritance in PASION", Simulation 50(6).

Raczynski S., 1990, "Graphical description and a program generator for queuing models", Simulation 55(3).

Raczynski S., 1991, "Simulacio'n por com putadora - metodos y lenguajes", LIMUSA.

Shmucker K. J., 1986, "Object-oriented Languages for the Macintosh", BYTE 11(8), August 1986, pp. 177-185.

Ziegler B. P., 1976, "Theory of Modeling and Simulation", John Wiley & Sons.

Author biography

Stanislaw Raczynski received his Ph.D in automatic control from the Academy of Mining and Metallurgy in Poland, 1969. He has been working for several years at the same Academy, at the Institute for Control Problems in Moscow USSR and at the National University of Mexico. Recently he is a professor at the Pan-American University in Mexico City. He is an active member of SCS, working in the area of control and simulation methods.

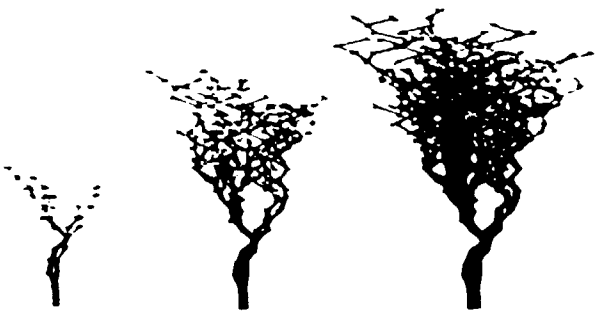


Figure 1. Simulation of a growing plant

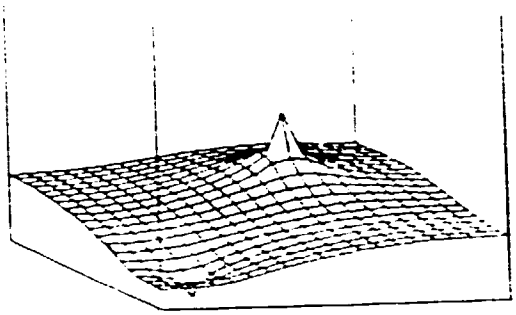


Figure 2. Heat transfer simulation

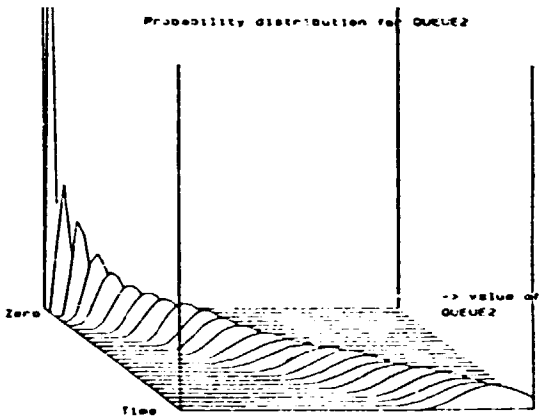


Figure 3. Example of a QMG output. The 3D image of the probability density function for a queue

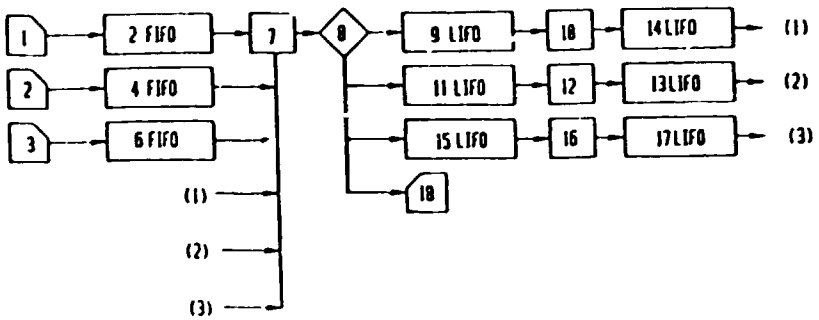


Figure 4. An example of a QMG scheme. 1, 2 and 3 are input streams, 2, 4, 6, 9, 11, 15, 14, 13, and 17 are queues, 7, 18, 12 and 16 are servers.

12. APPLICATIONS

The Australian research centre Csiro has introduced to the European market what it claims is the first computer program which simulates metal solidification in a pressure die-casting machine. Based on initial studies undertaken by the Battelle Institute in the USA, DMT-Castflow has been adopted by VAG in Germany and Norsk Hydro in Norway and in France Renault has been persuaded to use the program in designing an oil sump for the Clio model. (Source: Metal Bulletin Monthly, December 1991)

* * * * *

Simulator improves problem solving

Analogies are powerful language tools because they help us better understand abstract processes and interactions. Software for Apple Macintosh computers, called I Think, uses an analogy of tanks, pipes, and valves for modelling almost any kind of process.

The term process is used in a broad sense. It can refer to the interpersonal dynamics of a team, flow of a product through piping, or a strategic plan. The analogy is useful because a broad range of people may participate when designing a model. The result is an operational picture that graphically conveys the inner workings of an organization.

Placing icons in a diagram window builds the process. Rectangles represent anything that accumulates in a human process, such as man-hours, cash, inventory, and even frustration. The most common rectangles, called stock, resemble a tank with a continuously changing level. Stock representing finished goods, for example, fills as production finishes, and drains with shipments. Value of the stock at any time corresponds to inventory level.

There are also icons for discrete process accumulations such as conveyor belts and queues. A pipe and valve icon portrays flows which fill and drain accumulations, such as revenues, expenses, cash, and shipments. Circle icons are converters for logical relationships and equations (Profit = Revenues - Expenses) which transform a set of values into another.

Other converters transform values through a graphical relationship. For instance, a graph can describe how productivity decreases with work time. Even when an equation is unknown, the graphical converter permits sketching a curve with the mouse. An arrow tool provides plumbing that connects the diagram together. The software also has more than 50 built-in functions.

While the user constructs a process, the software creates a system of simultaneous equations. The package requires no programming language. Users can select one of three simulation algorithms (Euler's method, 2nd-order Runge-Kutta, and 4th-order Runge-Kutta), and adjust the simulation step size.

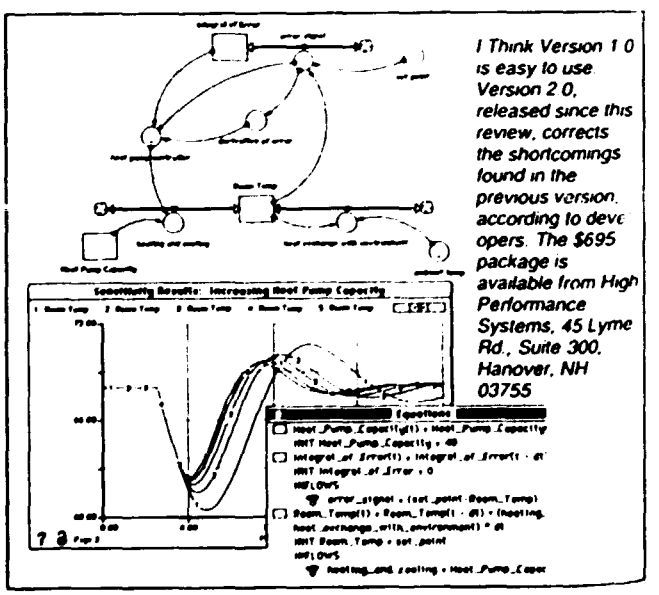
Just creating the model provides insight to a process. Creating a picture to accurately depict the process forces you to think about how it works, which is a big step in solving the problem.

Unlike a static spreadsheet, models are visual and dynamic. An animation feature lets you see reservoirs filling and draining, while flow variations show as gauge movements on valves. This helps impart an intuitive feel for the interdependencies among the elements. Even feedback loops, usually difficult to construct in a spreadsheet, are handled easily.

Plots available include time series, comparative, and scatter. Tables capture numerical values for detailed evaluations of model runs. A sensitivity-analysis compatibility allows running a model several times by assigning different increments and ranges to variables.

Documentation does a good job. Theory is well explained and users may adapt several generic models. Nevertheless, modelling is a skill and requires a degree of expertise in developing the proper structures.

The package had a few shortcomings. For instance, model diagrams, graphs, and tables can be printed but not directly annotated. Printing equations as a separate list is acceptable, but printing them directly on the diagram would be helpful when trouble-shooting a complex model. (Source: Machine Design, 21 November 1991)



I Think Version 1.0 is easy to use. Version 2.0, released since this review, corrects the shortcomings found in the previous version, according to developers. The $695 package is available from High Performance Systems, 45 Lyme Rd., Suite 300, Hanover, NH 03755

Simulation: Hands-on training without the risks

"Practice makes perfect", but learning from mistakes in industry can be costly. In the steel industry, mill operators and design and maintenance engineers can now rely on advanced simulations of plant processes and equipment to gain valuable experience without the risks of disrupting production.

Many engineers and operators in the steel industry learn their process knowledge direct from their peers and supervisors, by working alongside them in the field. Formal training is usually only given when a new process or piece of equipment is introduced; even then, training is often restricted to the knowledge immediately required for the workforce to do its job, without providing a sufficient background understanding of the process and products involved.

The reduction in manning in the steel industry over the last 15 years has particularly affected the R&D departments, often leading to the loss of a generation of engineers with the allround process knowledge and experience. Companies are therefore turning to outside consultants for specialist training and assistance for design and justification of new process and equipment upgrades.

Simulation as a training tool

Mathematical models and simulations of the rolling process have been available for many years. Typically, buried away within R&D departments, they are often poorly documented and unfriendly to users without maths or computer science qualifications. Recognizing the potential of such models as valuable training tools, the UK based Broner Group and Industrial Automation Services (IAS) of Australia have specifically developed basic models to provide user-friendly packages that are accessible to a wider audience of engineers. Verified with data from a variety of steel and aluminium mills, these models can provide "true-to-life" simulator training.

Dynamic Mill Simulator

The most recent development is the Dynamic Mill Simulator for multi-stand cold mills, with up to six stands. All phases of rolling, from threading to de-threading, including control loop closure, acceleration and deceleration can be simulated. Interactions between thickness, tensions and shape are all included. Different control structures can also be "installed" for comparison and analysis of behaviour. Running in real-time, on a PC, trainees can use the simulator to develop their skills offsite. The screen displays mimic those found in a real mill control pulpit and the provision of a "control desk", with all the controls available to a mill operator, concludes the true-to-life scenario.

Simulation in training courses

Models and simulations are an integral part of the International Rolling Technology Course (IRTC), held annually for the rolling industry. Used for practical tutorial sessions, a full range of rolling mill analysis and simulation software allows trainees to test and put into practice the theories learnt in the lectures. It has long been recognized that the quickest, most effective way of learning is "to do", as well as listen.

The intensive, 5-day course is aimed specifically at filling the gap in training available to R&D and mill engineers. Through a comprehensive combination of lectures and hands-on study the IRTC covers the complete rolling process, from the detailed mathematics of the roll gap process, to the design of a full automation system and the effects of operating practices on mill throughput.

Simulation for consultancy

Simulations are also being recognized as valuable consultancy tools and are used in projects to design equipment parameters, to assess proposals for equipment upgrades, and to justify capital expenditure. Invaluable in evaluating different control systems and investigating mill behaviour, the models have been used to develop thickness, tension and shape control schemes and full mill automation systems, now installed and implemented on a number of mills world wide. With the installation of these new systems, mill behaviour may be different and new operating practices may be required, but through the use of a simulator, operators and engineers can learn about these changes beforehand.

The concept of simulator training is of course not new; the flight simulator, developed to train pilots without risk to their passengers and crew, is perhaps the most well known example. The major risks from trainees in a steel mill are more from the cost in time and money of machine down-time, loss of production and poor quality products. If simulations can allow operators and engineers to test their skills and designs prior to implementing them, then the application and development of simulators for the steel industry is an invaluable investment for the future.

For further information on Rolling Mill Simulation, the International Rolling Technology Course and other training and consultancy services, please contact The Broner Group Limited U.K. on: Tel.: (0) 923-777619. Fax: (0) 923-773431. (Source: Steel Times, January 1992)

13. PUBLICATIONS

Material science and technology: A publication of The Institute of Materials, Vol. 8, No. 2, ISSN 0267-0836. The February 1992 issue is a "special issue" devoted to **MODELLING OF MATERIALS**. The articles were presented at a meeting held on 9 April 1991 at the University of Cambridge and organized by the Institute of Metals as an integral part of "Metals and materials '91".

* * * * *

Whicker, Marcia L. & Sigelman, Lee. Computer Simulation Applications: An Introduction. (Applied Social Research Methods Ser.: Vol. 25). (Illus.). 160 p. 04/1991. $28.50. (ISBN 0-8039-3245-6); Paper. $13.95. (ISBN 0-8039-3246-4). Sage Publications, Incorporated.

* * * * *

Landau, D. P.; Mon, K. K. & Schuttler, H. B., editors. Computer Simulation Studies in Condensed Matter Physics III. 224 p. 09/1991. $64.00. (ISBN 0-387-53607-8). Springer-Verlag New York, Incorporated.

* * * * *

Nersesian, Roy L. Computer Simulation in Financial Risk Management: A Guide for Business Planners & Strategists. 04/1991. $47.95. (ISBN 0-89930- 578-4, NCD, Quorum Bks). Greenwood Publishing Group, Incorporated.

* * * * *

McHaney, Roger. Computer Simulation: A Practical Perspective. (Illus.). 288 p. 08/1991. (ISBN 0-12-484140-6). Academic Press, Incorporated.

* * * * *

Schriber, Thomas J. An Introduction to Simulation Using Gpss-H. 01/1991. Hardcover text edition. $44.95. (ISBN 0-471-04334-6). Wiley, John & Sons, Incorporated.

* * * * *

Thavikulwat, P. Management Five Hundred: A Computer Simulation for Business Strategy & Policy. 02/1991. Incl. IBM disk. $15.16. (ISBN 0-07- 834713-0). McGraw-Hill Publishing Company.

* * * * *

Birdsall, C. K. & Langdon, A. B. Plasma Physics Via Computer Simulation. (Adam Hilger Plasma Physics Ser.). 502 p. 06/1991. $70.00 t incl. disc. (ISBN 0-7503-0117-1, Inst. Physics UK). American Institute of Physics.

* * * * *

Fishwick, P.; Luker, P. A. & Schmidt, B., editors. Qualitative Simulation Modeling & Analysis. (Advances in Simulation Ser.: Vol. 5). (Illus.). 360 p. 04/1991. Paper. $49.00. (ISBN 0-387-97400-8). Springer-Verlag New York, Incorporated.

* * * * *

Haug, E. J. & Deyo, R. C., editors. Real-Time Integration Methods for Mechanical System Simulation. (NATO ASI Series F: Computer & Systems Sciences: Vol. 69). viii, 352 p. 02/1991. $79.00. (ISBN 0-387-53280-3). Springer-Verlag New York, Incorpoated.

* * * * *

Simulation Symposium (Twenty-Fourth Annual). LC 71-149514. 348 p. 1991. $60.00 (ISBN 0-8186-2169-9, 2169). IEEE Computer Society Press.

* * * * *

Simulators V (ESC 1988, Orlando). (Simulation Ser.: Vol. 19, No. 4). Date not set. $40.00. (ISBN 0-911801-34-0). Society for Computer Simulation.

* * * * *

Tools for the Simulation Profession (ESC 1988, Orlando). Date not set. $28.00. (ISBN 0-911801-36-7). Society for Computer Simulation.

14. PAST EVENTS AND FUTURE MEETINGS

1992

31 March - 2 April The Hague	UTECH '92 - Conference and Processing Seminars on: - Elastomer Technology; - Coatings, Adhesives, Sealants and Encapsulants; - Flexible Foam Technology; - Rigid Foam Technology; and - Microcellular Foam Technology. (Crain Communications Ltd., Cowcross Court (2nd Floor), 75-77 Cowcross Street, London EC1M 6BP. Fax: +44 (0) 71-608-1173)
6-8 April Bordeaux, France	The Fibre Society Meeting - first time in Europe with the support of: EACM (European Association for Composite Materials) Bordeaux and ITF (Institut Textile de France) Lyon (Dr. J. Skelton, Albany International Research Co., 777 West St., P.O. Box 9114, Mansfield, MA 02048-9114, USA. Fax: 508-339-4996)
7-9 April Gamisch-Partenkirchen, Germany	Magnesium Alloys and their Applications (Deutsche Gesellschaft für Material-kunde e.V. Adenauerallee 21, 6370 Oberursel 1, Germany)
7-10 April Bordeaux, France	5th European Conference on Composite Materials (European Association for Composite Materials (EACM), 2 Place de la Bourse, 33076 Bordeaux Cedex, France)
8-10 April Thessaloniki, Greece	3rd International Conference on Energy and Building in Mediterranean Area (Laboratory of Building and Construction Physics, Dept. of Civil Eng., Aristotle University of Thessaloniki, P.O. Box 429, 54006 Thessaloniki, Greece. Fax: (031) 200392)
9-14 April Oaska, Japan	JP'92 - Plastics and Rubber Fair (JP Fair Association, Ginza Yamagishi Bldg., 2-10-6 Ginza, Tokyo 104, Japan)
13-16 April St. Louis, MO, USA	Magnetics and Magnetism and Magnetic Materials (Institute of Electrical and Electronic Engineers, 655 Fifteenth Street, Suite 300, Washington, D.C. 20005)
24-28 April Berlin, Germany	4th World Congress on Biomaterials (Institute of Pathology, Steglitz Clinic, Free University of Berlin, Hindenburgdamm 30, 1000 Berlin 45)
13-15 May Düsseldorf, Germany	The Recycling of Metals (ASM European Office, rue de l'Orme, 19 Olmstraat, B-1040 Brussels, Belgium Fax: 322733-43-84 - 734-67-02)
17-22 May St. Louis, MO, USA	High TcSuperconductor Technologies (Electrochemical Society, 10 Main St., Pennington, NJ 08534)
18-21 May Orlando, FL. USA	5th Int. Conf. on Creep of Materials (ASM International, Materials Park, OH 44073, USA)
9-11 June Stockholm, Sweden	Int. Conf. on Stainless Steel 92 (The Institute of Metals, 1 Carlton House Terrace, London SW1Y 5DB)
11-12 June London, UK	Rubber & Eastern Europe 1992 (First Europe Communications, 16 Treadgold St., London W11 4BP)
14-17 June Vancouver, Canada	Hydrometallurgy Theory and Practice (University of British Columbia, Dept. of Metals and Materials Engineering, 309-6350 Stores Rd., Vancouver, B.C. V6T 1W5 Canada)
15-17 June Paris, France	Automation in Fatigue and Fracture Testing and Analysis (Société Française de Metallurgie, Immeuble Elysees La Defense, Cedex 35, 92072 Paris La Defense, France)
21-26 June San Francisco, CA, USA	First World Congress of Powder Metallurgy Bridging Borders to Particulate Materials (Metal Powder Industries Fed., 105 College Road East, Princeton, NJ 08540-6692, USA)
22-24 June Paris, France	Workshop on Plastic Optical Fibres and Applications (IGI Europe, c/o AKM AG, P.O. Box 6, CH-4005 Basel, Switzerland. Fax: 41-61-691-8189)
22-26 June Thessaloniki, Greece	6th Int. Conf. on Intergranular and Interphase Boundaries in Materials (University of Thessaloniki, Dept. of Physics, 54006 Thessaloniki, Greece)
22-27 June Trondheim, Norway	3rd Int. Conf. on Aluminium Alloys (ICAA3, SINTEF Metallurgy, N-7034, Trondheim, Norway)
23-27 June Hangzhou, China	1st Pacific RIM International Conf. on Advanced Materials and Processing (PRICM-1) (Sponsoring the four-day conference are The Minerals, Metals & Materials Society (TMS), Chinese Society of Metals (CSM), Japan Institute of Metals (JIM) and Korean Institute of Metals (KIM)) (PRICM-1 Secretariat, Chinese Society of Metals, 46 Dongsixi Dajie, Beijing, People's Rep. of China. Fax: 86-01-5124122)
24-26 June Cambridge, MA, USA	Electronic Materials (The Minerals, Metals and Materials Society, 420 Commonwealth Dr., Warrendale, PA 15086, USA)
28 June - 2 July San Diego, CA, USA	7th World Conference on Titanium (Japan Institute of Metals, Aoba Aramaki, Aoba-ku, Sendai 980, Japan)

28 June - 3 July Manchester, UK	Advances in Corrosion and Protection (Corrosion & Protection Centre, UMIST, P.O. Box 88, Manchester M60 1QD, UK)
3-8 July Beijing, China	2nd Symposium on Physics of Magnetic Materials (San Huan R/D Center, Academia Sinica, P.O. Box 603, Beijing 100080, China)
12-15 August Gothenburg, Sweden	EASST MEETS 4S IN GOTHENBURG Annual Meeting of the Society for Social Studies of Science (4S) jointly with the European Association for the Study of Science and Technology (EASST). The 1992 conference has two preliminary themes: - "500 years after Columbus" - "Europe after 1992" (Center of Science Studies, Gothenburg University, S-412 98 Gothenburg, Sweden. Fax: +463163-47-23)
1-6 September Coeur d'Alene ID, USA	Microstructures (ASM International, Materials Park, OH 44073, USA)
7-10 September Birmingham, UK	Interdisciplinary Research Conference Materials for High Performance (University of Birmingham, IRC in Materials for High Performance Applications, Edgbaston, Birmingham B15 2TT)
7-11 September Berlin, Germany	Third International Conference on Low Cycle Fatigue and Elasto-Plastic Behaviour of Materials (Organized by Deutscher Verband für Materialforschung und -prüfung (DVM), held under the auspices of the Fed. of Europ. Materials Soc. (FMS) and co- sponsored by a number of int'l bodies including The Institute of Metals). DVM-Office, Unter den Eichen 87, W-1000 Berlin 45, Germany. Fax: (030) 811-93-59
7-11 September Kyoto, Japan	2nd Iketani Conference on Defusion in Materials (DIMAT '92, Dept. of Metal Science and Technology, Kyoto University, Sakyo-ku, Kyoto 606, Japan)
8-10 September Tokyo, Japan	1992 PAN-Pacific Pulp and Paper Technology Conference (Canadian Pulp and Paper Association, 1155 rue Metcalfe, Montreal, Quebec, Canada H3B 4T6)
8-10 September Amsterdam, Holland	Composites ECCM Testing & Standardization (EACM, 2 Place de la Bourse, 33076 Bordeaux Cedex, France)
22-25 September Yokohama, Japan	2nd International Conference on Computer Applications to Materials and Molecular Science and Engineering (The Nikkan Kogyo Shinbun Ltd., Kudan-kita-1, Chiyoda-ku, Tokyo 102)
29 September - 2 October Tokyo, Japan	6th International Conference on Ferrites (Tokyo Institute of Technology 2-12-1, Dept. of Physical Electronics, Okayama, Meguro-Ku, Tokyo 152)
27-31 September Hannover, Germany	Euro-Blech '92 - 12th Int. Sheet Metal Working Technology (Mack-Brooks Exhibition Ltd., Forum Place, Hatfield, Herts. AL10 ORN, UK)
3-6 November Madrid, Spain	4th European Electric Steel Congress (CONGRHISA, CEE '92, Velaquez, 90-5, 28006 Madrid, Spain)
16-17 November Miami, FL, USA	Compression Response of Composite Structures (ASTM 1916 Race St., Philadelphia, PA 19103-1187)
17-20 November Kyoto, Japan	Heat Treatment of Materials (Research Institute for Applied Science, 49 Tanaka Ohi-cho, Sakyo-ku, Kyoto, 606 Japan)
30 November - 1 December Solihull, West Midlands, UK	Biennial International Conference on Plastics and Rubber in Automotive Technology (Plastics and Rubber Inst., 11 Hobart Place, London SW1W 0HL, UK. Fax: +44 (0) 71-823-1379.